НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

До захисту допущено:
Завідувач кафедри
_____Олександр. РОЛІК
«___»_____20___р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Персональний сервіс автоматизованого навчання на базі месенджеру Телеграм»

Виконав:
студент IV курсу, групи ІА-62
Мокалець Дмитро Михайлович _____

Керівник:
Ст. викладач
Яланецький Валерій Анатолійович _____

Рецензент:
_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр РОЛІК
«___»_____20__р.

ЗАВДАННЯ

на дипломний проєкт студенту

Москальцю Дмитру Михайловичу

1. Тема проєкту «Персональний сервіс автоматизованого навчання на базі месенджеру Телеграм», керівник проєкту Яланецький Валерій Анатолійович ст. викладач, затверджені наказом по університету від «07» травня 2020р. №1081-с

2. Термін подання студентом проекту 09.06.2020

3. Вихідні дані до проєкту: Формат теоритичного матеріалу – PDF презентації. Структура тестів з однозначними відповідями. Зберігання результату у спеціально розробленій формі 1, інтерфейс програми – базовий діалог у месенджері Telegram. Программа розроблена на мові C#. Тест розроблений на мові C#.

4. Зміст пояснювальної записки: Вступ, аналіз предметної області, огляд існуючих рішень, дослідження сервісів для автоматизації навчання, технічна реалізація програми та її тестування

5. Перелік графічного матеріалу: схема класів, схема функціональна, схема логіки програми, схема прецедентів.
6. Дата видачі завдання 30.04.2020

Календарний план

№	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд та аналіз предметної області	01.05.2020 – 03.05.2020	
2	Пошук технологій для розробки системи	03.05.2020 – 05. 05.2020	
3	Порівняння існуючих рішень	05.05.2020 – 07.05.2020	
4	Проектування структури системи	08.05.2020 – 11.05.2020	
5	Розроблення структурної схеми	11.05.2020 – 15.05.2020	
6	Аналіз якості та тестування програмного забезпечення	17.05.2020 – 20.05.2020	
7	Оформлення текстової документації	20.05.2020 – 01.06.2020	
8	Подання готового проєкту	09.06.2020	

Студент Дмитро МОСКАЛЕЦЬ

Керівник Валерій ЯЛАНЕЦЬКИЙ

АНОТАЦІЯ

Мокалець Д.М. Телеграм-бот для автоматизованого навчання, КПІ ім. Ігоря Сікорського, Київ, 2020.

Об`єкт дослідження: чат-бот для перевірки знань студента.

Мета проекту: розробка автоматизованого сервісу який зможе надати студентові матеріал з предмету та перевірити його знання за допомогою тестування.

У першому розділі було проведено аналіз предметної області дипломного проекту.

У другому розділі було проведено огляд існуючих рішень по створенню сервісів для автоматизації навчання.

У третьому розділі було описано патерні для розробки програми.

У четвертому розділі було описано алгоритм реалізації чат-боту для мессенджеру Telegram та його тестування.

У додатках наведено: код програми, діаграма автоматизиції у навчанні, список тесту.

ANNOTATION

Mokalets DM Telegram bot for automated learning, KPI. Igor Sikorsky, Kyiv, 2020.

Object of research: a chatbot to check the student's knowledge.

The purpose of the project: development of an automated service that will be able to provide the student with material on the subject and test his knowledge through testing.

In the first section the analysis of the subject area of the diploma project was carried out.

The second section reviews existing solutions for creating services to automate learning.

The third section described the patterns for program development.

The fourth section described the algorithm for implementing a chatbot for the Telegram messenger and its testing.

The appendices contain: program code, learning automation diagram, list of test.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА62.160БАК.005 ПЗ	Пояснювальна записка	61		
6	А3	ІА62.160БАК.005 Д3	Персональний сервіс	1		
7			автоматизованного навчання на			
8			базі телеграм. Діаграма прецедентів.			
9						
10	А3	ІА62.160БАК.005 Д2	Персональний сервіс	1		
11			автоматизованного навчання на			
12			базі телеграм. Діаграма			
13			функціональності			
14						
15						
16	А3	ІА62.160БАК.005 Д3	Персональний сервіс	1		
17			автоматизованного навчання на			
18			базі телеграм. Діаграма класів			
19						
20	А3	ІА62.160БАК.005 Д4	Персональний сервіс	1		
21			автоматизованного навчання на			
22			базі телеграм. Діаграма логіки			
23			бота			
24						
25						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА62.160БАК.005 ТП		
Розроб.		Москалець.Д.М.			Персональний сервіс автоматизованого навчання на базі месенджеру Телеграм. Технічний проект	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Яланецький В.А				Т	
Реценз.						КПІ ім. Ігоря Сікорського ФІОТ група ІА-62	
Н. Контр.							
Затв.							

Пояснювальна записка

до дипломного проєкту

на тему: «Персональний сервіс автоматизованого навчання на базі месенджеру Телеграм»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПК – персональний комп’ютер;

ПЗ – програмне забезпечення;

ПДР – правила дорожнього руху;

Бот – програма, що виконує певні дії задані користувачем через інтерфейс який надає сама програма;

Чат-бот – бот який працює в середині месенджера, соціальній мережі, основною функціональністю якого є ведення діалогу з користувачем;

Telegram — популярний месенджер, який дозволяє обмінюватися текстовими повідомленнями та різноманітними файлами;

Месенджер – програма для обміну повідомленнями або файлами між користувачами;

ПК – персональний комп’ютер;

C# – об’єкто – орієнтована популярна мова програмування для платформи .NET.

ВСТУП

В наш час під час стрімкого розвитку технологій, автоматизаційних процесів, швидкого обміну інформації з'являються все більше речей для полегшення життя людини. Значну частину роботи взяли на себе роботи.

З'явилися перші роботи під кінець 20-х років. З того часу почали стрімко розвивати свою функціональність та швидко увірвались на світовий ринок. Не обійшло це стороною і популярні месенджери, які надають для користувачів більший вибір функціоналу додаючи до своїх програм чат–ботів.

Для автоматизації навчального процесу було обрано чат–бота який перевіряє знання з певного предмета. Важливою проблемою під час мого навчання в університеті була незручність повторення матеріалу перед важливою контрольною або заліком. Всі методички по певному предмету в месенджерах було проблематично знайти оскільки різних повідомлень в тому числі і файлів було забагато. Тому я вирішив, що для полегшення цих проблем слід додати до месенджера мого бота, який не тільки дає методичку, а ще й перевіряє знання тим самим показуючи на скільки студент готовий до пари.

Тут дуже важливою складовою є авторизація користувача. Оскільки основна суть бота полягає не тільки полегшити життя студентові а і викладачеві. Коли у викладача немає часу опитати всіх студентів він може оцінити їх знання завдяки боту який видає ім'я, фамілію, та результат тестування. Це дуже важливий момент для оптимізації всього навчання, оскільки перехід на віртувальне опитування студентів є еталоном розвитку всього університету. Не кажучи вже про користь для довкілля, оскільки більше не потрібно буде використовувати зошити.

Бот представляє з себе спеціалізований додаток, який базується на платформі обміну повідомленнями, дозволяючи користувачами взаємодіяти зі сторонніми сервісами, через знайомий інтерфейс чату. Тобто, для того, щоб отримати певну інформацію, людині не потрібно залишати межі месенджера,

досить відправити спеціальну команду, яка відповідним чином інтерпретується зі сторони бота. Наприклад, можна швидко отримати інформацію про ту послугу на якій спеціалізується бот.

Чат–боти – це віртуальні помічники, які зазвичай вбудовуються в месенджери та мають можливість розмовляти з реальними людьми. Їх головна функція – це вміння вести діалог з користувачем, і відповідно до функцій, закладених розробником, вони можуть отримувати, оброблювати, надавати певну інформацію. Вони можуть стежити за деякими подіями, отримувати та надсилати данні про погоду, можуть вести облік витрат та доходів, можуть знаходити різного роду інформацію, і звичайно просто покращувати настрій користувачеві.

З моменту появи влітку 2015 року, кількість ботів для вирішення найрізноманітніших завдань, зростає вражаючими темпами. З їх допомогою вже можна також оперативно отримати інформацію про погоду, курс валют, новини, перекладати слова і фрази, розважатися, обмінюватися файлами, і вирішувати безліч інших завдань.

Суть створення чат–боту для месенджерів базується на використанні відкритого API запитів. Клієнтом може бути як основна програма месенджера так і його мобільна версія. Реалізацією ідеї може бути як сервіс, що створено власноруч так і об’єднання вже існуючих рішень з використанням їх переваг. Те що обрав користувач має взаємодіяти з ним, отримуючи повідомлення від нього, формувати та відправляти запит на сервер. Сервер, отримавши цей запит, обробляє його, та відправляє текст користувача на NLP сервіс, який повинен обробити текст, повернути дані в придатному для обробки вигляді назад на сервер.

Для створення бота дуже важлива взаємодія з користувачем. Тому потрібно максимально забезпечити його невеликим та зрозумілим вибором функцій. Потрібно пам’ятати що чим менша його функціональність тим швидша та ефективніша його робота.

Зараз створення чат-ботів є досить актуальною темою в повсякденному житті, оскільки потреба в їх функціональності дедалі зростає. Не кажучи вже про багато вакансій на біржі праці, де зачасту можна побачити замовлення боту для якоїсь кампанії (бот для банку, бот для замовлення піцци, бот для прогнозу погоди на тиждень).

Найголовніше що розвиток цієї сфери буде завжди стрімко зростати вгору, бо її потреба буде актуальною завжди.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Автоматизація навчання

Галузь автоматизації дозволяє модернізувати будь - яку роботу в житті людини . Це представляє собою автоматичне(без людини) впливи на технічні процеси в повсякденному житті. Це може бути навіть банальний комп’ютер, який ми всі використовуємо. Те саме стосується і навчання.

Автоматизація навчання робить навчання будь якої галузі науки більш сучасною. Це можуть бути чи доступ до навчального матеріалу чи реєстрацією на ЗНО чи банальна перевірка знань за допомогою тестування. Для автоматизації навчання робиться все аби полегшити доступ для перевірки знань чи то індивідуально чи то перевіряючи якийсь контроль знань.

Було розглянуто такий вид автоматизації навчання як сервіс для надання тестів для перевірки знань з певної теми. Використовував я для цього найпопулярніший в Україні месенджер – Telegram. Для цього месенджера існує спеціальний додаток – телеграм бот. Суть бота полягає в тому, аби виконувати його функціонал , тобто це окрема програма вбудована в програму.

Сама концепція цієї роботи полягає в тому аби зробити бота, який проводититиме тест з з певного предмету, який добавить користувач.

В першу чергу розробник має орієнтуватися на викладачів та студентів політехнічного інституту, які зможуть осучаснити вивчення матеріалу .

Зі сторони викладача :

– перевірка тестів студента;

– розробка власного тесту для опитування за допомогою методички прелставленній в навчальній програмі університету;

– отримання бази даних зі повним списком результатів тесту серед студентів;

Зі сторони студента :

— змога перевірити власні знання;

— змога отримання швидкої оцінки без очікування своєї черги на сдачу матеріалу;

— доступ цілодобово до боту, що дозволяє у будь – який момент запустити тест;

Головною моєю задачею була розробка коду та створення тесту з вибіркової теми я вибрав оцінку знань з програмування на мові C#.

1.2 Створення тестування

Для автоматизації навчання було вирішено зробити сервіс тестування у месенджері Telegram. Сервіс буде розроблятися на мові C#.Головною нашою задачею є реалізація боту, який буде отримувати дані які ввів наш користувач. Також розробка тесту з продуманими питаннями по вибраній темі. Потрібно також розробити форму для доступу до ресурсу, завдяки якій користувач адмін зможе змінювати питання та створити новий тест.

Особливістю нашого тесту буде те, що він буде проходитися у самому телеграмі. Але для подальшого розуміння потрібно розуміти як взагалі працює чат – бот.

1.3 Телеграм боти

Чат–бот компютерна програма, розроблена на основі нейромереж та технологій машинного навчання яка веде розмову за допомогою слухових або текстових методів.

Чат–бот імітує розмову з людиною в інтернеті, саме тому цей сервіс найкраще зарекомендував себе у таких помулярних месенджерах як Facebook та Teletgram.

За допомогою таких ботів клієнт може вирішити проблему без оператора, не чекаючи черги на обслуговування потреби.

Слід додати про актуальність ботів. Зараз по статистиці більшість людей надають перевагу месенджерам ніж таким популярним програмам як Youtube, Twiter, Instagram. Це зв'язано з тим, що в месенджерах ви можете використати ті функції, які надають все вищеперечісленні. Завдяки легкому доступу та малому споживанні пам'яті на телефоні або на ПК користувачі надають перевагу їм .

Тому стало очевидно, що набагато простіше знайти до користувача там, де він проводить більшість свого часу, а не створювати нові застосунки, і намагатися нав'язати їх йому. Також чат-бот набагато простіше запустити, через те що його не потрібно скачувати та встановлювати на пристрій, не займає місця в пам'яті, та на дисплеї пристрою.

Зазвичай ботів використовують для автоматизації рутинних дій, які вони можуть виконувати самостійно або допомагати в пошуку інформації, в перекладі, відстежувати певні події, відповідати на повідомлення, проводити опитування, взаємодіяти з датчиками та речами, які підключені до інтернету, зберігати данні, нагадувати про певні події, вбудовуватися в інші сервіси і платформи, тобто вони можуть робити все, що їм буде запрограмовано відповідно до інструментів розроблення платформи.

Але і є певні недоліки. На жаль на данний момент бот все одно не зможе замінити нормальну людину як співрозмовник. Більшість людей будуть спілкуватися з чат–ботом просто із–за цікавості що він тобі відповість далі. Будуть по всякому його тестувати тим самим забираючи у себе дрогоцінний час. Це повязано з тим що люди не звикли до взаємодії подібного роду співрозмовника. Беручи до уваги більшість літніх людей де для того щоб пояснити як користуватися ботами знадобиться досить багото часу. Але слід додати що розвиток йде на зручність користування, тому через деякий час користування ними навіть старим людям буде не проблема.

1.4 Команди ботів

Як виглядають команди ботів. Команди є загальним способом виклику певної події від боту.

Команди викликаються по певній ознаці такій як: / Команда

Перша команда завжди викликається автоматичино. Це команда :

/start. Це дія запускає нашого бота для подальшого користування.

Взагалі кожна команда починається з символу косої риски «/» та має довжину до 32 символів. Команди використовують букви латинського алфавіту, цифри та підкреслення.

Приклад простого використання популярного бота навчального-бота(@ZnoTestsBot):

Бот просить ввести предмет з якого ви б хотіли пройти тестування. Перед цим він видає список свїх предметів. Я наприклад хочу пройти тестування з української мови. Ввожу в діалог слово «тест». Потім ми бачимо, що нам надається список предметів , які користувач може вибрати для тестування. Наприклад тестування з математики (/id301) та тестування з української мови(/id318). Бачимо особливість цього списку в тому, що до кожного предмету прикріплені посилання в душках, такі званні посилання при виборі конкретного видає результат вибранного нами предмету. Для цього достатньо прописати команду вибраного предмету. Пам’ятаємо що кожне звернення до бота починається з символу «/». Далі вибираємо ай-ді рецепту який прописаний у дужках кожного навчального предмету. Наприклад я хочу пройти тест з української мови, тому прописую:

/id_318

Після команди бот її швидко обробляє та видає список питань з вибраного предмету або посилання на сайт з тестуванням цього предмету, в залежності

від того як запрограмований наш чат-бот.

1.5 Процес створення телеграм-боту

На відміну від взаємодії з ботом, його створення є справою не такою вже й легкою. Для цього використовується мова програмування, оскільки, по суті, йдеться про програму, яка приймає команди зі сторони користувачів, і працює на основі Telegram Bot API. Створення ботів в телеграмі майже для всіх однакове, для цього потрібно виконати декілька пунктів і бот буде створено:

1) викликати головного бота BotFather;

2) після цього прописати команду /newbot;

3) після цього обрати назву для бота;

4) обрати адрессу бота для користувача;

5) отримати токен від Botfather:

6) вставити токен в свій програмний код;

Якщо у програмному коді не виявлено жодних помилок бот буде працювати.

У чат – боті дуже візуально не вдосконалиш свою програму, але можна вибрати аватар бота , назву , та різний його опис. Аватар можна вибрати будь – який, тут немає цензури. Аватар підтримує найпопулярніші формати зображень, тому не буде проблемою брати картинку зі сторонніх ресурсів. Також для додатку роботи програми можна зробити автоматичне надсилання картинок користувачеві бота. Повідомлення буде приходити користувачеві для реалізації своїх графічних потреб. Слід додати що неважливо як графічно виглядає бот оскільки ми повинні пам’ятати що чат – бот не є окремою програмою а тільки доповненням.

Для повної готовності його використання потрібно задіяти в самому меседжері Телеграм. Використовуючи при цьому певні команди про які треба довідатись у самого бота.

1.6 Правила користування ботами

Усе, що потрібно для роботи з ботами, – акаунт у Telegram. Для вас взаємодія з ними виглядатиме як спілкування в чаті, різниця лише в тому, що на іншому кінці не людина, а програма з початками штучного інтелекту.

У роботів немає статусів «онлайн» і «був у мережі», натомість відображається напис «робот». Окрім цього, боти не можуть самі почати спілкування. Обраного бота треба спершу додати в групу або першим почати з ним діалог. Для цього можна використовувати посилання виду telegram.me/ <ім'я бота> або пошук за іменем користувача.

Знайти ботів просто, оскільки в них ім'я завжди закінчується на «bot».

Щоб запустити бота треба ввести команду : /start

Щоб дізнатись функціонал бота ввести команду : /help

1.7 Функціональнівсть ботів тестувальників

Під час отримання завдання розробити бота тестувальника знань я одразу згадав моє навчання в автошколі, де кожному учню надається методичка та тести по цій методичці. Хороші боти тестувальники зазвичай працюють по тій самій схемі.

Відбувається авторизація учня або студента після проходження тесту видає результат (скільки правильних відповідей з тесту було виконано правильно). Така, здалося б, банальна функціональність дуже оптимізує навчання в багатьох онлайн школах. І загалом всі навчальні курси намагаються додати до своєї програми такого робота.

Висновки до розділу

Під час аналізу предметної області було встановлено, що бот – це програма, яка автоматизує певні процеси (отримання та обробка даних, вчинення відповідних дії до отриманих даних, проведення процесу створення чого-небудь тощо).

Одним з видів ботів є чат-боти. Чат-боти зазвичай використовуються для різного роду роботи з інформацією (збір даних з опитувань, побудова відповідних графіків, надання консультації замість реальних людей, відсилання сигналу щодо певних змін в певній системі тощо). Чат-боти дуже легко вбудовуються в системи, де відбувається процес спілкування з користувачем (соціальні мережі, месенджери, інтернет ресурси в яких відбувається процес спілкування).

Також ми розглянули приклади використання тест ботів у сфері освіти. Продивилися їх функціональність, їх створення, та взаємодію з ними.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1Огляд сервісів автоматизації навчання

Знаючи що моя мета це зробити сервіс автоматизованого навчання я перебрав багато варіантів для вирішення своєї проблеми. Я вирішив зробити бота, який буде проводити тестування користувача в режимі онлайн.

Сама концепція навчання для мене це перевірка знань з певної області. Зачасту контроль знань проводиться в писемному виді в університеті чи в школах, що є не дуже зручно, оскільки витрачає багато часу на перевірку результату. Викладач перебирає всі відповіді кожної зданної роботи, коли міг би просто виписати оцінку з комп’ютера, який сам би все перевіряв за декілька секунд. Тому мені потрібно створити щось подібне до так званного електронного додатку вбудований в меседжер Telegram.

Автоматизоване навчання застосовує інформаційні технології, які дозволяють зробити процес навчання більш індивідуальним та без допомоги викладача чи вчителя. Тут забезпечується оперативний самоконтроль та діагностика помилок. Саме тому я зрозумів, що створити онлайн тест у такому популярному меседжері як Telegram буде гарною ідеєю. Сама концепція онлайн тесту потрібна для забезпечення взаємодії між користувачем та викладачем. Викладач самостійно робить тести з його предмету, а студент проходить їх. Знаючи по собі коли проходиш будь - який тест декілька разів ти запам’ятовуєш інформацію на все життя.

Є також певні недоліки такого сервісу. Викладач ніколи не буде виставляти оцінку за такий самоконтроль оскільки не буде впевненний у чесності студента.

Я розглянув декілька схожих ботів до моєї теми в плані функціоналу. Всі боти можна оглянути в Telegram, знайшовши їх за вказаною назвою та додавання в кінці Bot.

2.1.1Бот ZNOUAPollBot

ZNOUAPollBot – бот для опитувань, який був створенний для викладачів шоб перевіряти знання учнів з того чи іншого предмета.

Особливість цього бота полягає в тому що тут основна задача це взаємодія тільки з викладачами, оскільки його функціональність призначена для створення опитування, що є дуже корисним для розробки сервісу автоматизованого навчання. Суть одної з програм якраз полягає у тому, щоб за допомогою сторонніх методичок додавати тестування за смаком. Це може бути тестування на будь – яку тему.

Бот має основні доступні команди:

– /help – відображає меню користування ботом;

– /newpoll – створює

– /peview – переглядає створене опитування;

– /finish – опубліковує створене опитування;

– /start – розпочинає спілкування з ботом;

– /terminate – видаляє опитування;

– /password – викладач отримує пароль;

Також особливість цього бота це те що опитування можуть створити лише викладачі. Для цього надається пароль та при початку роботи з ботом ввести його тоді коли бот попросить ваш пароль.

Цей бот дуже корисний для навчання та перевірки знань своїх учнів перед ЗНО.

Недоліки бота : обмежене використання для учнів.

Якщо порівнювати з сервісом який треба розробити то він тільки на половину підходить по функціоналу, оскільки є головна задача взаємодії з викладачами які будуть додавати свої тести для опитування.

2.1.1Бот AndyEnglishBot

Всіх людей які люблять саморозвиток об’єднує одне – це бажання вивчити нову мову. Так сталося що для найпопулярніша мова для навчання в світі це англійська. Люди вчать слова, проходять курси у різних школах. Проходять тестування і закріплюють результат. Більшість професій потребують знання англійської мови. Також вона потрібна для спілкування з людьми, які живуть за кордоном. Знання англійського це один з найголовніших навичок у світі для взаємодії з навколишнім світом. Та скільки ти не витрачав часу на вивчення слів та проходження тестів найголовніше для цього навику є практика спілкування.

Саме для цього був розроблений такий чат - бот як AndyEnglishBot. Його функціональність полягає у спілкуванні з користувачем на англійській мові.

Приклад використання AndyEnglishBot:

Користувач після вводу команди “/start” отримує від бота повідомлення для початку діалогу :

– Hi my name is Andy

Після чого користувач може вступити в контакт з ботом. Бот має широку бібліотеку слів для взаємодії з юзером. Він може спитати як його настрій чи дізнатись про навички користувача. Якщо користувач відповів неправильно то може застосувати свою функцію автокорекції тим самим навчаючи правил граматики англійської мови.

Однак у кінці після декількох речень бот повідомить вам про те, що це лімітована версія:

– It was limited demo version. You should download my app for Android.

Це і є головним недоліком програми.

Як ми бачимо бот має непогане розуміння того що пише користувач у діалозі. Дуже вражає і те що бот володіє навичком автокорекції та у змозі виправити свого співрозмовника, що є дуже важливим при вивчанні граматики англійської мови.

Але головним недоліком : спілкування з ботом має лімітовану версію і для подальшого використання потрібно заплатити.

2.1.3 Бот ZNOGrantBot

Незважаючи на назву ЗНО тут проводиться опитування не тільки по ньому. Тут надається користувачеві вибір : почитати матеріал чи пройти тестування по ньому. При чому тут не обов’язково прописувати команду через символ «/». Оскільки робот розуміє і звичайну мову а також цифри(можна відповісти йому 1 або 2).

Також його особливістю є те, що він кожного дня надсилає певний матеріал який дуже знадобиться для ЗНО. Матеріал оформенний дуже компактно для прочитання. При виборі прочитання методички або тесту бот дасть посилання на відповідний сайт.

На мою думку саме подібним чином і повинен виглядати кожен бот тестувальник, оскільки тут немає нічого зайвого, окрім надсилання кожного дня нового матеріалу(повинна бути команда, яка виключає цю функцію). Це займає пам'ять вашого ПК або мобільного.

Головним недоліком : відсутня команда виключення присилання додаткового матеріалу тим самим займаючи зайву пам'ять комп’ютера або мобільного телефона.

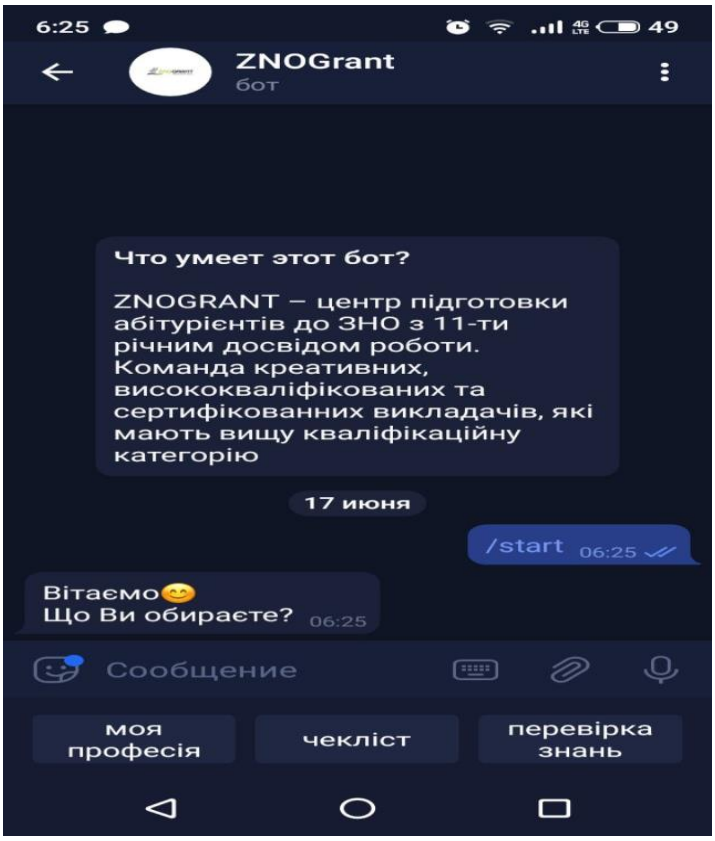


Рисунок 2.1 — Робота ZnoGrant бота

Але з позитивного окрім надачі інформації та тестів тут є і посилання на консультацію з приводу майбутньої професії, що дуже важливо для будь-якого школяра завчасно визначитися зі своїм майбутнім. Поради завжди дуже важливі, бо одною з основних проблем суспільства є те що учень який навіть випускається з 11 класу є досить незрілою особистістю. Йому завжди потрібна правильна консультація з цього приводу.

2.1.2Бот InMindBot

Дуже корисний бот для вивчення і перевірки знань з англійської мови. Його функціонал полягає в тому щоб кожен день взаємодіяти з користувачем. Вам будуть присилати повідомлення про те, що вже час перевіряти свої знання рисунок 2.1.

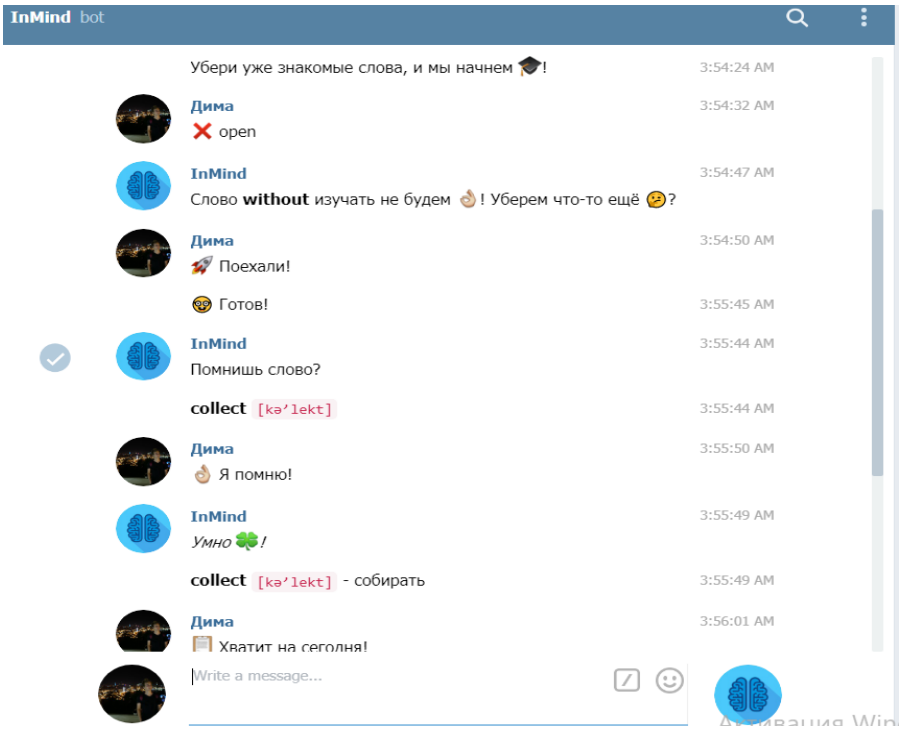


Рисунок 2.2 — Робота InMind бота

Коли ми починаємо працювати з ботом він дає нам 5 слів на вибір, та просить вибрати ті, які нам знайомі. Після того як ми виконали його прохання починається тестування. Він дає нам так званний диктант на який ми маємо відповісти переклад слова з англійської. В кінці він видає наш бал. Слід сказати що окрім перекладу слів він надає користувачеві його транскрипцію, що є дуже важливим при вивченні англійської та правильного граматичного спілкування. Але є і суттеві недоліки. Як це дивно не звучало б, але його дуже складно удалити зі свого чату. Навіть після виделення він продовжує присилати повідомлення.

2.1.5 Бот AvtoshkolaUzBot

Тест бот для автошколи є досить невід'ємною частиною для навчання і перевірки знань учня. Для надачі матеріалів цей бот використовує посилання на Youtube відео уроки або відкриття методички.

Загалом функціональність цього боту дуже проста , оскільки має досить

малий запас команд.

Приклад команд:

/start – запуск бота;

/off – зупинити підписку;

Основною взаємодією між користувачем і ботом є кнопки посилання які є дуже зручні для будь - якого юзера. Потрібно пом’ятати, що до автошколи йде не тільки молодь а і люди яким треба пересдавати на права. Люди схильного віку не дуже розуміють що значить прописувати боту команду. Саме тому бот має дуже зручний інтерфейс.

Кнопки використання відповідають на умови:

— відеоуроки;

— нравила дорожнього руху за 2016 рік;

— найпопулярніші питання в тестах;

— зв'язок з адміністрацією;

Як бачимо бот дуже зручний та зрозумілий по функціональності. Але він має суттєві недоліки.

Недоліки : ПДР — не теперішнього року.

Висновки

Усі боти дуже корисні для оптимізації навчання, але як і всі програми вони мають свої недоліки.

Більш за все до моєї роботи ближчий це ZNOGrantBot. Я можу запозичити для соєї ідеї багато функціональності з нього. Але мені не підходить функція надсилання додаткового матеріалу кожні 24 години.

У таблиці 2.2 наведено порівняльний аналіз ботів.

Таблиця 2.2 – Порівняльний аналіз телеграм ботів

Критерій оцінки	ZNOUA PoolBot	AndyEnglishBot	ZNOGrant Bot	InMind BoT	Avtoshkola UzBot
Корисність	+	+	+	+	+
Гнучкість (вміння розпізнавати контекст повідомлення)	-	+	-	+	-
Простота використання	-	-	-	+	+
Автоматизація процесу	+	+	+	+	+
Достатність інформації	-	-	+	+	+

Бачимо що з усіх ботів найкраще за всіх бот InMind, так як він володіє непоганою взаємодією з користувачем, має бібліотеку для надачі відповідного матеріалу користувачеві, має збалансовану та небаговану систему опитування англійської мови та вбудовані кнопки програми які поширюють його функціонал.

2.4 Вибір додаткового сервісу для реалізації

2.4.1Asp.NET core

Платформа ASP.NET Core представляє технологію призначену для створення різного робу веб – додатків, від невеликих веб – сайтів до великих веб – порталів і веб – сервісів.

Розробник як правило може працювати поверх крос – платформного середовища net.Core, яке може буди розгорнуте на основних популярних операційних системах такі як: Windows. Mac OS. Linux. І таким чином, за допомогою ASP.NET Core розробник може створити платформні додатки. Тобто ми можемо запускати веб – додатки не тільки на Windows. Але і на Linux Mac ОС. А для розгортання веб – додатків можна використовувати традиційнай, або крос – платформний веб – сервер.

Крім об'єднання вищезазначених технологій в одну модель було додано ряд додаткових функцій.

Однією з таких функцій є тег – хелпери, які дозволяють більш органічно поєнувати синтаксис з кодом C#.

ASP.NET Core характеризується поширюваністю. Фреймворк побудований з набору незалежних компонентів. І розробник може або використовувати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або зовсім створити і заспадкувати компоненти зі своїм функціоналом.

Також було спрощенно управління зілежностями і основною конфігурацією проекту. Фреймворк тепер має свій легкий контейнер для впровадження залежностей, і більше немає необхідності застосовувати сторонні контейнери.

2.4.2 Azure

Microsoft Azure – це набір хмарних служб, який допомагає вам вирішувати бізнес – завдання . Це свобода створення, розгортання додатків управління ними в великій глобальній мережі з використанням інструментів і платформ.

Працездатність платформи Microsoft Azure забезпечує глобальна мережа розподілених дата-центрів Microsoft.

Для нас головним є те, що базові функції операційних систем, Microsoft Azure має і додаткові: виділення ресурсів на вимогу для масштабування, автоматичне синхронну реплікацію даних для підвищення стійкості, обробку відмов інфраструктури для забезпечення постійної доступності. Саме те що потрібно для створення нашої програми.

Модель надання інфраструктури реалізує для нашого бота можливість оренди таких ресурсів, як сервери. Це будуть наші пристрої зберігання даних. Управління всією інфраструктурою здійснюється постачальником, споживач управляє тільки операційною системою і встановленими додатками. Тобто той хто володіє структурою зазвичай іменують адміністратором сервера, він має більші привілегії такі як:

- управління сервером;
- використання додаткових інструментів;
- додачу нових файлів на сервер;
- update сервера;

В нашому випадку головною метою адміністратора це додати до своєї програми новий тест. Практично всі сервіси Microsoft Azure мають інтерфейс взаємодії API, побудований на основі обмежень для розподілених систем, що дозволяє мені як розробнику використовувати хмарні сервіси з будь-якою операційною системи, пристрої і платформи.

Крім того, усі користувачі можуть створювати і управляти власними сервісами, користуючись візуальним веб-інтерфейсом порталу Azure. Портал дозволяє налаштовувати сервіси, редагувати права доступу та відстежувати стан ресурсів.

Основна функціональність роботи з управлінням данних Microsoft Azure:

1) сервіс пошуку Azure забезпечує текстовий пошук;
2) redis Cache - це керована реалізація;
3) storSimple управляє сховищем і розподіляє навантаження між

локальними пристроями і хмарним сховищем;

4)SQL Database призначена для створення і розширення додатків в хмарі з використанням технології майкрософт сервер;

5)azure SQL Data Warehouse - це хмарне корпоративне сховище даних, що використовує масову паралельну обробку для швидкого виконання складних запитів користувача;

6)azure Data Factory - це служба, яка дозволяє створювати робочі процеси в хмарі для організації і перетворення даних;

7)azure Data Lake - це служба зберігання і аналізу даних для задач, пов'язаних з великими даними;

8)azure Stream Analytics - це серверний движок обробки подій, який дозволяє користувачам розробляти і запускати аналітику в реальному часі для декількох потоків даних;

Також невід'ємною частиною Microsoft Azure є обмін даними. Для цього використовуються так звані шини. Шина в Microsoft Azure дозволяє нашим додаткам, які працюють в хмарі або на наших зовнішніх пристроях обмінюватися даними з самою Azure.

Шина Azure підримує три типи комунікації, такі як:

– центри подій (забезпечують массову передачу данних в хмару. Найчастіше використовують для відстеження даних з мобільного телефону);

– розділи (Забезпечують зв'язок з використанням шаблону підписки між користувачем і додатком);

– ретранслятори (Забезпечують двосторонній зв'язок між сервером і користувачем)

Для реалізації нашого боту використовуємо данний додаткові сервіси для роботи нашого бота завдяки яким ми можемо додати на сервер нові тести для реалізації поставленого функціоналу.

Висновки до розділу

Під час огляду існуючих рішень було розглянуто п’ять різних ботів, було проаналізовано їх функціональність, основні переваги та недоліки, також було розглянуто два сервіси для того щоб зробити бота більш інтелектуальним.

Було виявлено, що боти є досить корисними застосунками, вони досить просто інтегруються в месенджер дозволяють автоматизувати ті чи інші завдання.

Основною задачею був розглядок інструментів для реалізації бота та плюси і мінуси п’яти ботів тестувальників. Ми виявили по декількох критеріям, а саме:

– корисність:
– гнучкість (вміння розпізнавати контекст повідомлення);
– простота використання;
– достатність інформації

Тестуючи їх отримали певні відомості про їх основний функціонал:

– взаємодіяти з користувачем;
– видавати правила користування;
– обробляти правильно потрібний запит;
– видавати вибраний користувачем функціонал(чи то надання методички, чи проходження тесту, чи спілкування на англійській мові);

Виявили що ідеальних ботів так само як і програм не існує. Тому повинні були вибрати найоптимальніший з приведенних ботів. Ним виявився бот з наданням інформації та надання змоги для тестування учнів.

Окремо можна сказати і про бота – перекладача , який себе вдало показав по всім критеріям, однак він мав доволі простий функціонал тому досить нечесно прирівнювати його до масштабного проекту реалізації правильного боту тестувальника.

Також ми розглянули додаткові сервіси для реалізації такі як:

- Asp.NET core
- Azure

Також виявили їх основний функціонал для допомоги нашої роботи. Дізнались те, що Asp.NET core потрібен нам для управління додатками та веб – сервісами, а Microsoft Azure допоможе нам з ними взаємодіяти за допомогою стандартних інструментів Microsoft. Крім того забезпечить нас інструментами для взаємодією з даними та збереження новостворенних сервісів для роботи власної програми.

3 ОГЛЯД ТЕХНОЛОГІЙ

3.1Паттерн Repository

Одним з найбільш часто використовуваних патернів при роботі з даними є патерн «Репозиторий». «Репозиторій» дозволяє нашій програмі абстрагуватися від конкретних підключень до джерел даних, з якими працює програма, і є проміжною ланкою між класами. Патерн безпосередньо взаємодіє з даними, і рештою програми.

Працює це по принципу заміни бази даних. Якщо у нас в один момент підключення до одної бази даних , а ми хочемо її змінити, то це буде не важкою задачею. При стандартному підході навіть в невеликому додатку, що здійснює вибірку, додавання, зміна та видалення даних, нам би довелося зробити велику кількість змін. Та в процесі роботи програми в залежності від різних умов ми хочемо використовувати два різних підключення. Таким чином, репозиторій додає програмі гнучкість при роботі з різними типами підключень.

Також особливістю є те, що наша система зі складною моделлю може бути спрощена за допомогою додаткового рівня, так званного Data Mapper, який працює в совокупності з нашим репозиторієм . У такому випадку може бути корисним додавання цього шару абстракції поверх нашого шару розподілу даних, в якому б був зібраний код створення запитів. Це стає ще більш важливим, коли в області визначення безліч класів або при складних, важких запитах. У таких випадках додавання цього рівня особливо допомагає скоротити дублювання коду запитів, що дуже оптимізує швидкість роботи нашої програми.

Патерн Repository є посередником між шаром області визначення і шаром розподілу даних, працюючи, як звичайна зберігаюча скриня об'єктів області визначення. Об'єкти, тобто клієнти створюють опис запиту і направляють їх до об’єкта сховища Repository для обробки. Об'єкти можуть бути додані або

видалені з сховища, як ніби вони формують просту колекцію об'єктів. А код розподілу наших даних, прихований в об'єкті Repository, подбає про відповідних операціях в непомітно для розробника.

У двох словах, патерн Repository інкапсулює об'єкти, представлені в сховищі даних, і операції, вироблені над ними, надають більш об'єктно-орієнтоване уявління реальних даних.

Але є і мінуси такого підходу. Якщо ви пишете більший проект ніж просто чат - бот, то кожен з об'єктів такого проекту не буде володіти своїм списком доступного функціоналу. Тобто може і бути невеличкий список , але основні функції будуть заблоковані. Наприклад метод видалення об'єкта, або додавання, може бути недоступний для цього типу об'єктів.

Висновок: даний патерн дозволяє мені як розробнику зручно працювати з даними мого проекта та, якщо є така необхідність в роботі проекта, міняти підключену до нього базу даних .

3.2Паттерн UnitOfWork

У додатках вже згаданного ASP.NET нерідко використовується патерн Репозиторій, про який ми описали до цього. Використовується для логіки роботи з джерелами даних та правильної заміни іншою базою данних. Приблизно так і виглядає наша робота. Нерідко ми працюємо з безліччю сутностей і моделей, для управління якими створюється також безліч класів – репозиторіів для більшої функціональності нашої програми. Тому нам потрібно додати патерн який буде взаємодіяти з усіма іншими патернами. Саме для цього ми будемо використовувати патерн Unit of Work, який дозволяє нам спростити роботу з різними репозиторіями і дає впевненість, що всі репозиторії використовуватимуть один і той же контекст даних. Патерн Unit of Work дуже тіснопов’язаний з патерном Репозиторій, оскільки один відповідає за зберігання данних(Репозиторій), а другий за їх правильну синхронізацію(UnitofWork).

Для того, щоб використовувати патерн Unit of Work, потрібно створити новий class UnitOfWork, який буде надавати доступ до репозиторіїв через окремі властивості і визначати загальний контекст, в нашому випадку для обох патернів: Repository, UnitOfWork

Крім того, даний клас містить додаткові методи Save () і Dispose (), які в іншій ситуації ми могли б визначити в репозиторіях, але так як цей функціонал буде загальним для обох репозиторіїв, то його краще винести в клас UnitOfWork.

Взагалі Unit of Work ,як ми вже говорили, це патерн, який визначає логічну транзакцію тобто синхронізацію змін в об'єктах, поміщених в об'єкт зі сховищем тобто базою даних(об’єкт Репозиторій). Якщо звернутися до вихідного опису цього патерну то видно що об'єкт, який реалізує цей патерн відповідає за накопичення інформації про те які об'єкти входять до транзакцію і які їхні змінити щодо вихідних значень в сховищі.

Також патерн Unit of Work як правило не є повністю самостійним, він зазвичай тісно пов'язаний з вбудованим в нього патерном Identity Map, завдання якого – збереження так званної карти створених об'єктів, взятих зі сховища. Робиться це для того щоб гарантувати що одиниця інформації зі сховища представлена рівно одним екземпляром об'єкта даних в додатку. Це дозволяє уникнути непередбачених змін тому не допускає ситуації коли два об'єкти представляють один і той же елемент даних в сховищі.

Підводячи підсумок: сам по собі Unit of Work досить простий в своєму зовнішньому інтерфейсі, але реалізація його коректної роботи вимагає надання безлічі додаткових даних, тому мініатюрних прикладів навести не можу.

3.3 Паттерн абстрактна фабрика

Основною задачею Абстрактної фабрики є надання для нас шаблону проектування. Також Абстрактна фабрика надає інтерфейс для створення сімейств взаємопов'язаних або взаємозалежних об'єктів. Шаблон реалізується створенням абстрактного класу Factory, який представляє собою інтерфейс для створення компонентів системи, наприклад, для віконного інтерфейсу він може створювати вікна і кнопки. Потім пишуться класи, що реалізують цей інтерфейс. Використовувати патерн Abstract Factory треба в двух випадках:

- якщо наша система повинна залишатися незмінною від процесу створення нових об'єктів.
- якщо нам необхідно створити групи взаємопов'язаних об'єктів виключаючи можливість їх одночасного використання

У нашому проекті ми повинні використовувати цей додаток для підтримки графічного інтерфейсу користувача. Це повинно розраховуватись для підтримки використання на різних платформах(Windows. Mac OS, Linux), при цьому зовнішній вигляд цього інтерфейсу повинен відповідати прийнятому стилю для тієї чи іншої платформи. Наприклад, якщо це програма встановлена на Windows платформу, то його кнопки, меню, смуги прокрутки повинні відображатися в стилі, прийнятому безпосередньо для Windows. Групою взаємопов'язаних об'єктів в цьому випадку будуть елементи графічного інтерфейсу користувача.

3.4 Паттерн Strategy

В проекті чат-бот існують системи, поведінка яких може визначатися відповідно до одного алгоритму з деякого сімейства класів. Всі алгоритми цього сімейства призначені для вирішення спільних завдань, мають

однаковий інтерфейс для використання і відрізняються тільки реалізацією , поведінкою. Користувач, попередньо налаштувавши програму на потрібний алгоритм (вибравши стратегію), отримує очікуваний результат. Як приклад додаток, призначений для компресії файлів використовує один з доступних алгоритмів:

- Zip;
- Arj;
- Rar;

Також в дизайн нашої програми може бути вбудованне використання поліморфізму. В результаті чого ми отримуємо набір родинних класів із загальним інтерфейсом і різними реалізаціями алгоритмів.

Суть нашого патерну це виділити потрібне з усіх наших класів. Загалом це деталі тісно пов’язані з реалізацією наших алгоритмів.

Також немало важливим в паттерні Strategy є вбудований класс Compressor, який потрібен для стиснення наших файлів(zip,rar,arj) породжуючи тим самим підкласи ZIP_Compression, ARJ_Compression і RAR_Compression. Він оголошує загальний інтерфейс для алгоритмів ztиснення. Завдяки цьому коли користувачеві потрібно перейменувати формат файлу в інший, алгоритм дозволить йому зробити це. Для заміни одного алгоритму іншим досить переналаштувати цей покажчик на об'єкт потрібного типу.

Як і у будь-якого партерна є свої переваги і недоліки в роботі програми.
Переваги паттерна Strategy:

1)Систему простіше підтримувати і модифікувати, так як сімейство алгоритмів перенесено в окрему ієрархію класів.

2)Патерн Strategy надає можливість заміни одного алгоритму іншим в процесі виконання програми.

3)Патерн Strategy дозволяє приховати деталі реалізації алгоритмів від клієнта.

Недоліки паттерна Strategy:

1) для правильного налаштування системи користувач повинен знати про особливості всіх алгоритмів.

2) число класів в системі, побудованої із застосуванням паттерна Strategy, зростає, тим самим зростає вага програми.

3.5 Структура SOLID

Паттерн SOLID розбитий на п’ять основних принципів об'єктно-орієнтованого програмування , яке описує архітектурну структуру програмного забезпечення конкретно для нашої програми.

Абревіатура SOLID розшифровується як :

S – принцип єдиної відповідальності;

O – принцип відкритості / закритості;

L – принцип підстановки Лісков, що описує можливості заміни примірників об'єктів;

I – принцип поділу інтерфейсів;

D – принцип інверсії залежностей;

Патерн SOLID допомагає мені як розробнику досягти масштабованості і поліпшити якість і надійність коду завдяки одночасному застосуванню принципів ООП. Системи, створені таким чином, щоб було просте обслуговувати, використовувати повторно і розширювати з плином часу, якщо знадобиться доповнити програму певним ширшим функціоналом.

Відповідно до принципу стандарту роботи програми, кожен клас повинен нести відповідальність за виконання тільки однієї речі. Це не означає, що в класі повинен бути тільки один метод, але, замість цього, всі методи повинні бути пов'язані однією метою. Даний стандарт дає можливість ідентифікувати класи на етапі дизайну програми. Перевага такого принципу з єдиною

відповідальністю полягає в тому, що зменшення розміру класів робить код простіше для розуміння, а самі класи стають легше в обслуговуванні і для повторного використання.

Якщо брати роботу програми за принципом відкритості / закритості то заміна поведінки класу буде в нас виконуватись за допомогою наслідування. Це означає, що проектувати наш клас потрібно таким чином, щоб додавання нового функціоналу не викликало труднощів при зміні вимог. Принцип закритості говорить, якщо клас вже розроблений і пройшов всі перевірки розробки, розробник не може змінювати код до тих пір, поки не будуть знайдені баги. Такий підхід також передбачає відкритість для розширення, але закритість для модифікації. Якщо клас використовується іншими, то він повинен бути відкритий для розширення. Щоб зробити його таким, потрібно додайти певне ключове слово в метод класу.

Принцип поділу інтерфейсу говорить про те, що клієнти не повинні залежати від методів, які вони не використовують. Замість одного інтерфейсу, краще використовувати групи методів. Інтерфейс повинен бути більш щільно пов'язаний з кодом, в якому він використовується, щоб методи інтерфейсу визначалися функціями, необхідними клієнтського коду. Чим більше інтерфейс, тим більша ймовірність, що він включає в себе методи, які можуть виконати не всі виконавці. І в цьому вся суть принципу поділу інтерфейсу. Інтерфейси потрібні тільки для досягнення низької пов'язаності між клієнтом та програмою.

Принцип інверсії залежностей говорить про те, що високорівневі модулі не повинні залежати від низькорівневих, тому, що ризик його поломки зростає. Тому, чим нижче зв'язаність між високорівневими і низькорівневими модулями, тим краще.

Висновки до розділу

В даному розділі ми розглянули п’ять основних паттернів для розробки автоматизованого сервісу. Було виявленно, що паттерн “Репозиторій” відповідає за взаємодію з даними проектуванню, яке дозволяє будувати оболонку програмию. Паттерн UnitOfWork дозволяє легко взаємодіяти з усіма іншими встановленими паттерними для роботи, що дозволяє працювати програмі правильно, та в випадку виправлення проблеми її можна легко без руйнування коду. Паттерн “Абстрактна фабрика” надає основний шаблон проектування, а паттерн Strategy надає стандартний функціонал для стиснення робочої програми до популярних форматів. Також зазначено, що паттерн SOLID застосовує пять основних принципів роботи ООП, що дозволяє коду стати більш надійним.

4 ТЕХНІЧНА РЕАЛІЗАЦІЯ

4.1 Загальна концепція ботів тестувальників

Суть полягає в перевірці знань за допомогою тесту. Саме тести допоможуть вдосконалити оцінювання знань студентів. Тепер викладачам набагато простіше проводити іспити, а також стежити за результатами і прогресом своїх учнів. Тепер вчителю або викладачеві не потрібно створювати кожен тест вручну і записувати результати в журнал, вираховувати середній бал. Сьогоднішні системи онлайн-тестування допомагають стежити за прогресом кожного учня, уникаючи складних підрахунків. Самі ж інструменти для проходження онлайн-тестів стають все зрозуміліше і зручніше як для учнів, так і для викладача.

У моєму чат ботові тестувальнику користувач зможе пройти тест на будь-який смак, все залежить від того у кого є адмінка. Коли у тебе є головний доступ ти можеш створювати тести самостійно. Особисто я для початку додав тест для оцінювання знань з програмування мови Сі-шарп.

Особливістю розроблених тестів є те що там можна створювати тести знову і знову на будь - який смак. Це може бути вищезгадана мова програмування, або тест з перевірки знань з автошколи, чи може звичайний тест на ерудованість. Я можу створити цілу базу запитань і зберегти їх, щоб використовувати в подальшому. Не потрібно писати окремий тест для кожного курсу. Я можу створити іспит просто вибираючи питання з уже створеної бази або скористатися шаблоном, який буде легко змінити для кожного нового курсу. Крім того дуже корисною виявляється автоматична система підрахунку балів.

Особисто вважаю що розроблений бот тесту вальний набагато переважає інші сайти з банальними опитуваннями, бо це зачасту недостовірне

оцінювання, яке базується на більш розвагах аніж достовірного результату.

4.2 Кодова реалізація

Для створення чат – боту була використана мова C#. Для початку нам потрібно за допомогу торренту додати до наших стандартних бібліотек програмування бібліотеку TelegramBot. Вона додає до нашого коду більшу функціональність задля створення бота для спеціальної мессенджер платформи. Ця бібліотека потрібна щоб не було помилки після звернення до класу telegram.message. Далі нам потрібно отримати токен для взаємодії нашої програми з месенджером Telegram.

Щоб отримати токен потрібно виконати такі дії:

1) викликати BotFather;
2) прописати команду /newbot ;
3) вибрати назву свого чат бота;
4) вибрати нікнейм для бота – тестувальника;

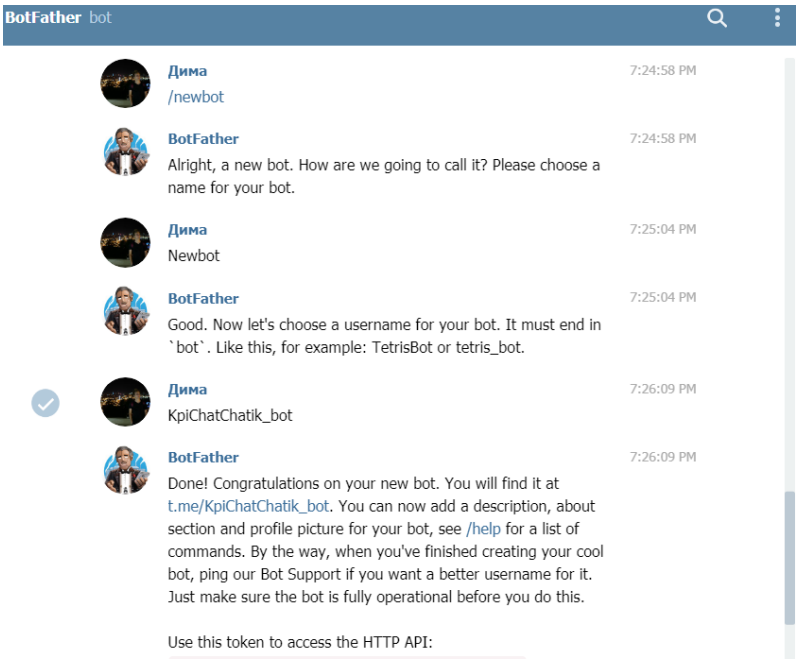


Рисунок 4.1 — Виклик BotFather

На рисунку 4.1 показана взаємодія з головним ботом мессенджера Teltgram.

Після вибору імені боту буде наданий токен доступа до бота за допомогою якого можна буде отримати доступ до боту, оскільки він є унікальним, і будь-хто, хто матиме токен доступу зможе використовувати бота, то значення його буде приховане.

Далі для статичної змінної приписуємо токен, який прислав нам BotFather. Після цього удосконалюємо код нашої програми. Головне для нашого бота для початку це запам’ятовувати всі повідомлення які відправляє йому користувач. Спочатку створюємо клас Program. Далі у класі додаємо метод static void main (string[]args). У цьому методі ми прописуємо команду яка за допомогою присвоювання буде зберігати всі повідомлення надіслані користувачем.

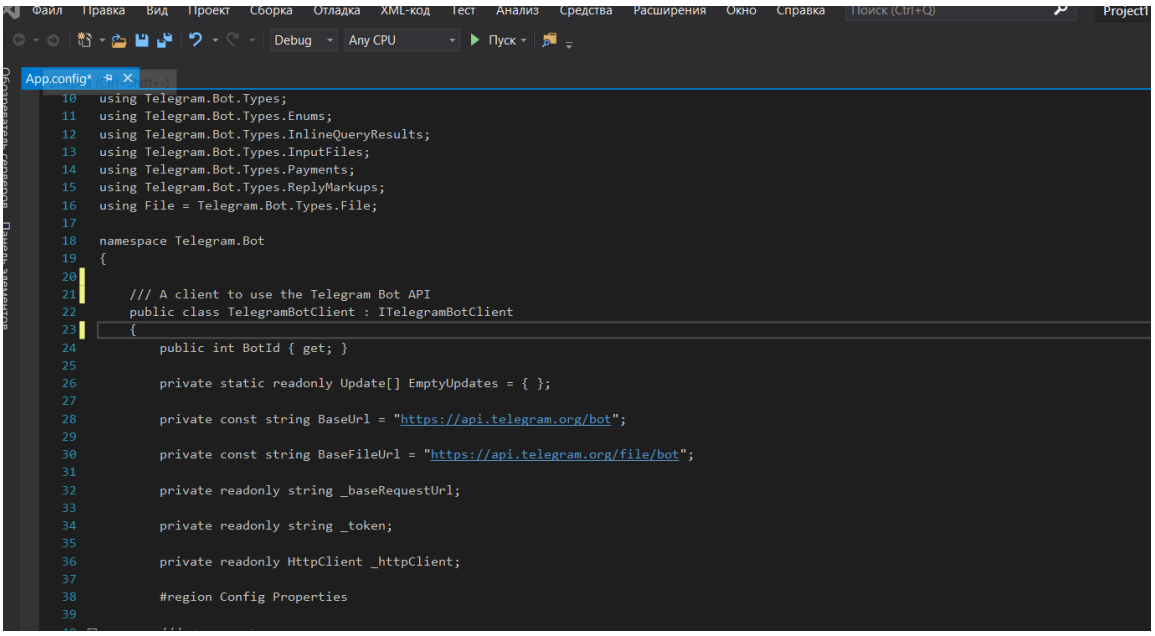


Рисунок 4.2 — Старт програми

На рисунку 4.2 зображено початковий код команди. Взаємодія з додатковою бібліотекою TelegramBot. Метод зберігань та відображення повідомлень.

Для перевірки роботи програми достатньо запустити її, надіслати до нашого бота повідомлення і воно буде відображатися у командній консолі програми C#.

Для подальшої реалізації функціоналу нашого бота потрібно додати ще декілька класів які будуть зберігати данні, робити запит нашому користувачеві та взаємодіяти з нашими токенами. Використовується такі класи окрім основної програми, як:

1) baseRequestUrl(зберігання даних);
2) httpClient(взаємодія з користувачем);
3)receivingCancel(отримання повідомлення та повторний запит вводу для користувача);
3) token(взаємодія з токенами);
4) EmptyUpdates(взаємодія зі змінами програми);

Кожний клас буде містити в собі властивості які дозволять програмі правильно працювати при вводі повідомлення користувачем.

Поміж цього до програми будуть приписані методи які ссилаються на посилання надане чат-ботом. Вони добавлятимуться до відповідних класів, а завдяки правильно прописаному програмному кодові легко взаємодіятимуть з додатковими бібліотеками та не видаватимуть помилку в програмі.

Дуже важливим що у бібліотеці TelegramBot є безпроблемний та швидкий спосіб реалізації таких функцій як передача аудіо, текстового файлу, команди ботів та навіть емодзі. Тому можна реалізувати і це, бо це знак хорошого тону для будь-яких ботів у телеграмі. Якщо сам мессенджер дозволяє користувачам спілкуватись з ботами подібним чином ми повинні зробити так, що при цьому не видавало помилку.

На даному етапі буде створена взаємодія з користувачем. Боту можна буде відправити повідомлення або прописати команду. Дуже важливим є здатність бота розуміти відправлене, бо при хибній команді користувача можна зробити повторний запит. Також тут працює і функція надання інформації взаємодії з ботом. Основною функцією є надання тесту для користувача. При виборі надання тесту для користувача буде видавати перший створений тест, який оцінює знання з програмування користувача мови C#. Сам тест був своренний в WPF, де реалізована паралельна програма для розробки нових тестів та надання адміністратора для неї. Сама программа буде надавати змогу створювати нову категорію тестів та вигадувати питання до вибраної теми.

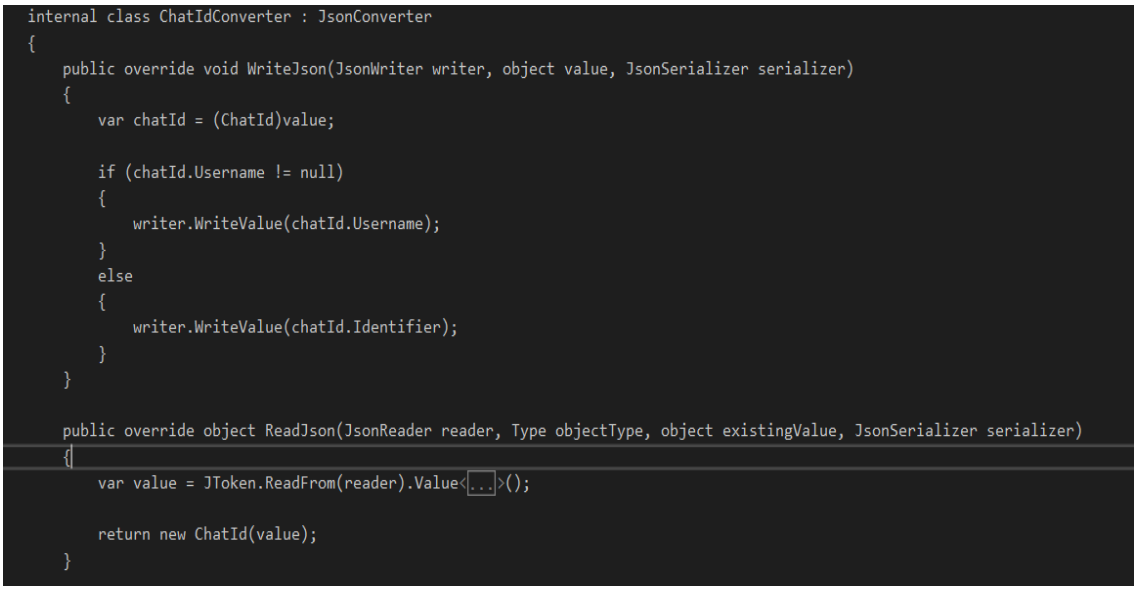
4.3 Перетворювачі програми

Для використання додаткового функціоналу в чат – боті ми дадамо до нашого коду конвертори, іншими словами перетворювачі. Кожен з них буде відповідати за певний функціонал такий як:

1) id повідомлення рисунок 4.3;
2) Вставка текстового файлу рисунок 4.4;
3) Вставка аудіо – файлу рисунок 4.5;
4) Розпізнавання типу повідомлення рисунок 4.6;

Основою будь-якого боту є його його початок діалогу з користувачем. Для того щоб вийти на id-адрессу з користувачем ми повинні реалізувати конвертор для вставки нашого точена, який нам вислав чат бот.

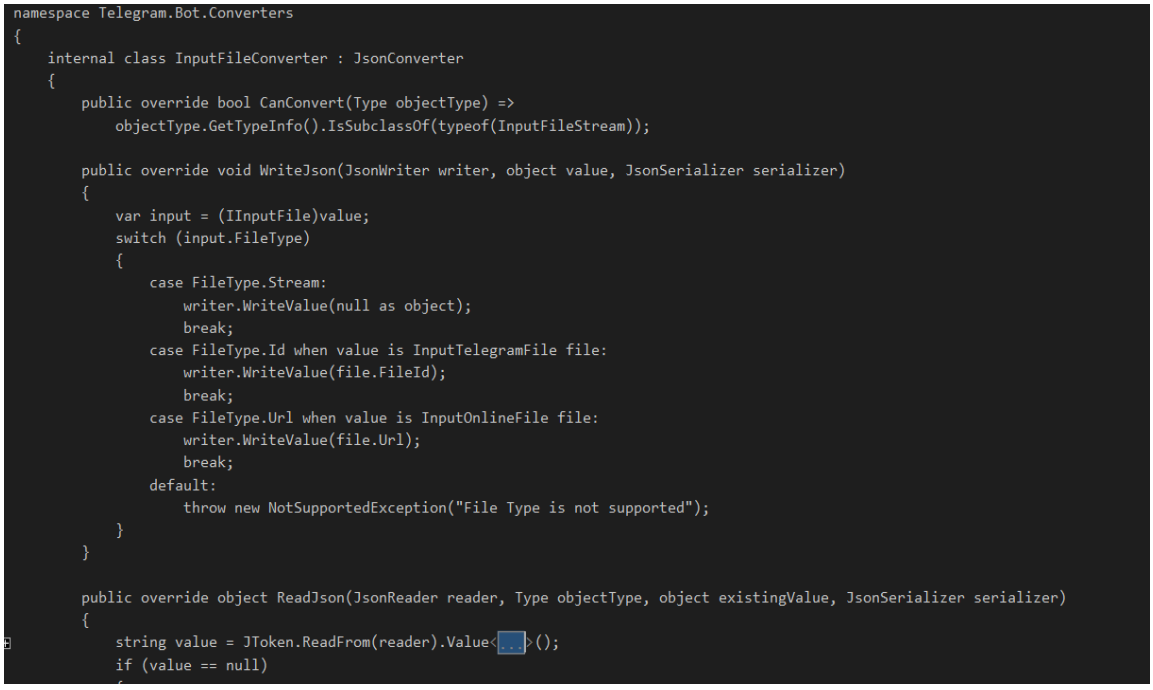
Даний конвертор потрібен нам для разпізнавання отриманого програмою повідомлення. Ми реалізуємо ситуацію якщо користувач не ввів своє ім’я, тобто не була проведена авторизація нового користувача, тоді наша програма впише нулюве значення.



```
internal class ChatIdConverter : JsonConverter
{
    public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
    {
        var chatId = (ChatId)value;

        if (chatId.Username != null)
        {
            writer.WriteValue(chatId.Username);
        }
        else
        {
            writer.WriteValue(chatId.Identifier);
        }
    }

    public override object ReadJson(JsonReader reader, Type objectType, object existingValue, JsonSerializer serializer)
    {
        var value = JToken.ReadFrom(reader).Value<...>();

        return new ChatId(value);
    }
```

Рисунок 4.3 — Перетворювач Id-повідомлення

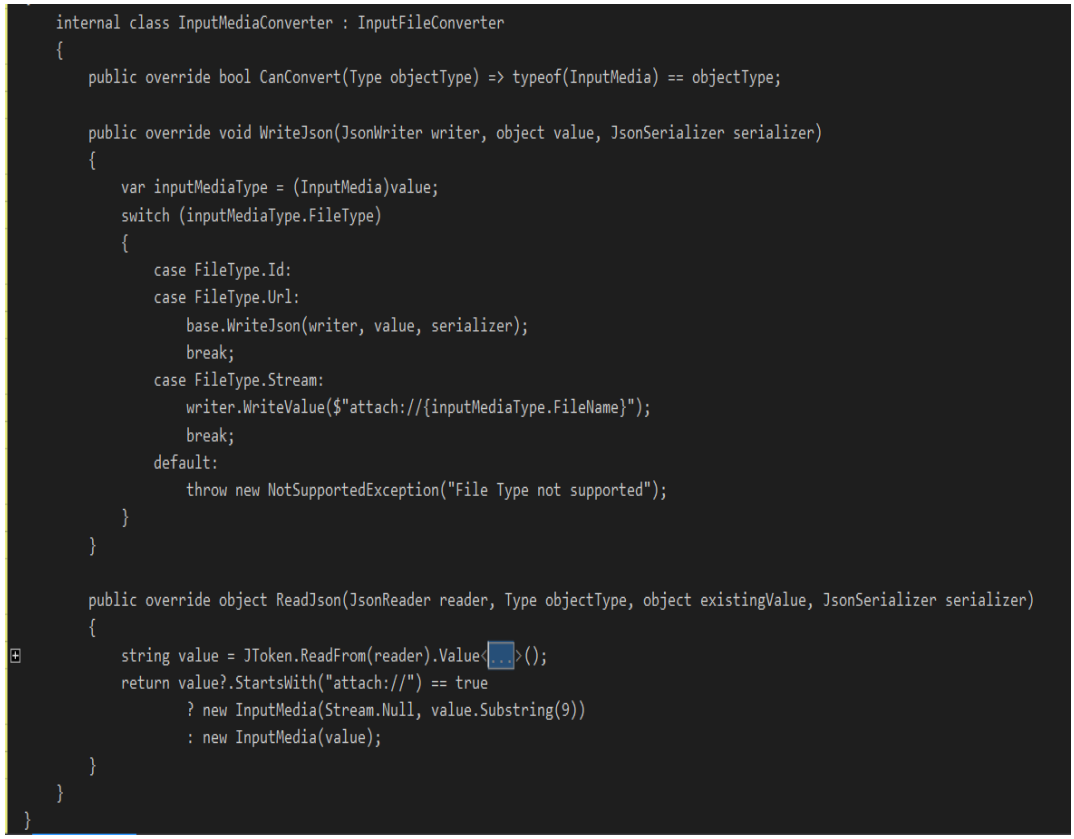
Для цього ми у кодовому значенні var value приписуємо токен нашого чат- боту.



```
namespace Telegram.Bot.Converters
{
    internal class InputFileConverter : JsonConverter
    {
        public override bool CanConvert(Type objectType) =>
            objectType.GetTypeInfo().IsSubclassOf(typeof(InputFileStream));

        public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
        {
            var input = (IInputFile)value;
            switch (input.FileType)
            {
                case FileType.Stream:
                    writer.WriteValue(null as object);
                    break;
                case FileType.Id when value is InputTelegramFile file:
                    writer.WriteValue(file.FileId);
                    break;
                case FileType.Url when value is InputOnlineFile file:
                    writer.WriteValue(file.Url);
                    break;
                default:
                    throw new NotSupportedException("File Type is not supported");
            }
        }

        public override object ReadJson(JsonReader reader, Type objectType, object existingValue, JsonSerializer serializer)
        {
            string value = JToken.ReadFrom(reader).Value<...>();
            if (value == null)
```

Рисунок 4.4 — Перетворювач текстового файлу

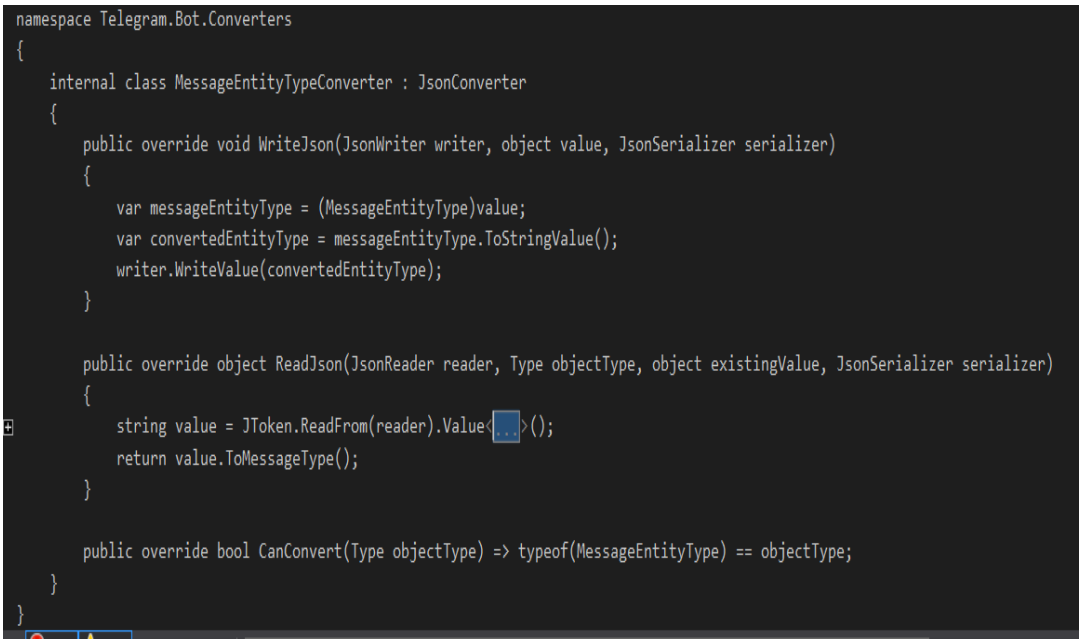
На данному рисунку показано кодову реалізацію для реалізації вставки текстового файлу в чат - боті.



```
internal class InputMediaConverter : InputFileConverter
{
    public override bool CanConvert(Type objectType) => typeof(InputMedia) == objectType;

    public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
    {
        var inputMediaType = (InputMedia)value;
        switch (inputMediaType.FileType)
        {
            case FileType.Id:
            case FileType.Url:
                base.WriteJson(writer, value, serializer);
                break;
            case FileType.Stream:
                writer.WriteValue($"attach://{inputMediaType.FileName}");
                break;
            default:
                throw new NotSupportedException("File Type not supported");
        }
    }

    public override object ReadJson(JsonReader reader, Type objectType, object existingValue, JsonSerializer serializer)
    {
        string value = JToken.ReadFrom(reader).Value<...>();
        return value?.StartsWith("attach://") == true
                ? new InputMedia(Stream.Null, value.Substring(9))
                : new InputMedia(value);
    }
}
}
```

Рисунок 4.5 — Реалізація взаємодії з аудіо – файлами

На данному рисунку показано кодову реалізацію для розпізнавання аудіо файлу в чат - боті.



```
namespace Telegram.Bot.Converters
{
    internal class MessageEntityTypeConverter : JsonConverter
    {
        public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
        {
            var messageEntityType = (MessageEntityType)value;
            var convertedEntityType = messageEntityType.ToStringValue();
            writer.WriteValue(convertedEntityType);
        }

        public override object ReadJson(JsonReader reader, Type objectType, object existingValue, JsonSerializer serializer)
        {
            string value = JToken.ReadFrom(reader).Value<...>();
            return value.ToMessageType();
        }

        public override bool CanConvert(Type objectType) => typeof(MessageEntityType) == objectType;
    }
}
```

Рисунок 4.6— Реалізація розпізнавання типу повідомлення

Та найголовнішою реалізацією було створена взаємодія користувача з чат – ботом за допомогою розпізнавання типу повідомлень на рисунок 4.5.

4.4 Методи

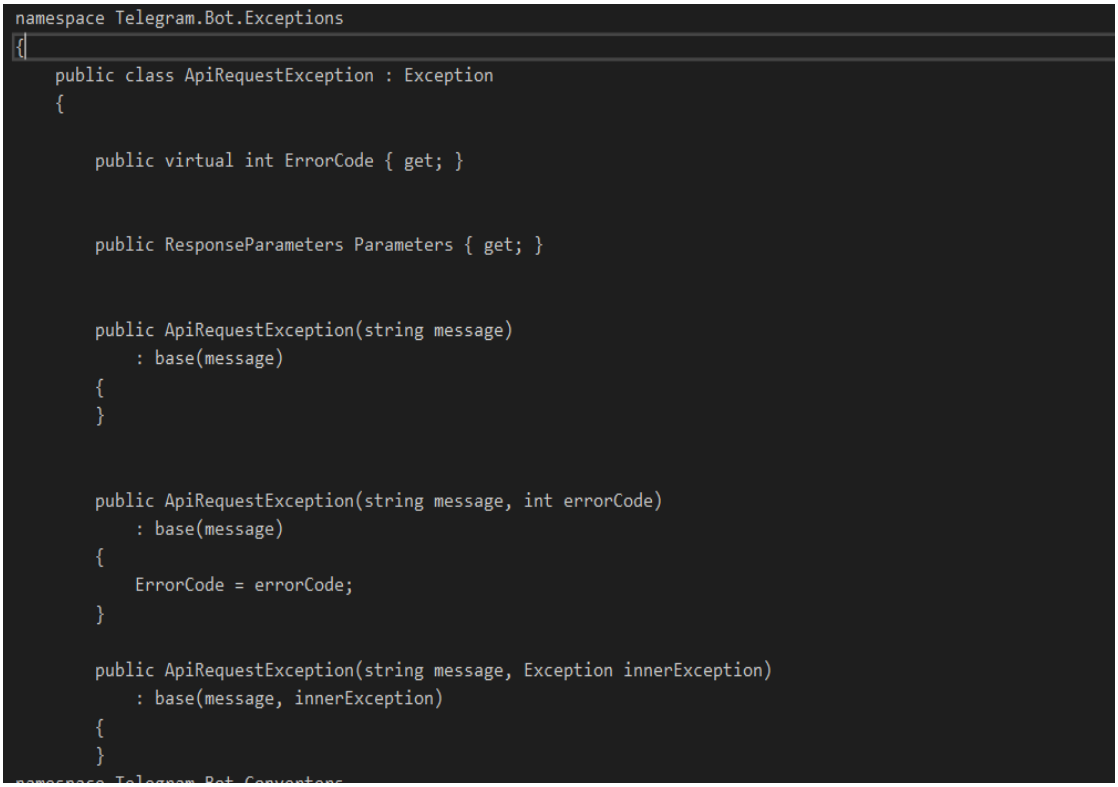
Методи в нас належать в основі реалізації нашого проекту. Кожен метод має свій персональний функціонал. Методи які використовуються для програми чат – бот:

1) Метод Args рисунок 4.6;
2) Метод Exeptions рисунок 4.7;
3) Метод Helpers рисунок 4.8;
4) Метод Requests рисунок 4.9;
5) Метод Types рис. рисунок 4.10;

```
namespace Telegram.Bot.Args
{
    public class MessageEventArgs : EventArgs
    {

        public Message Message { get; private set; }


        internal MessageEventArgs(Update update)
        {
            Message = (update.Type == UpdateType.EditedMessage) ? update.EditedMessage : update.Message;
        }


        internal MessageEventArgs(Message message)
        {
            Message = message;
        }
        public static implicit operator MessageEventArgs(UpdateEventArgs e) => new MessageEventArgs(e.Update);
    }
}
```

Рисунок 4.7 — Реалізація методу Args

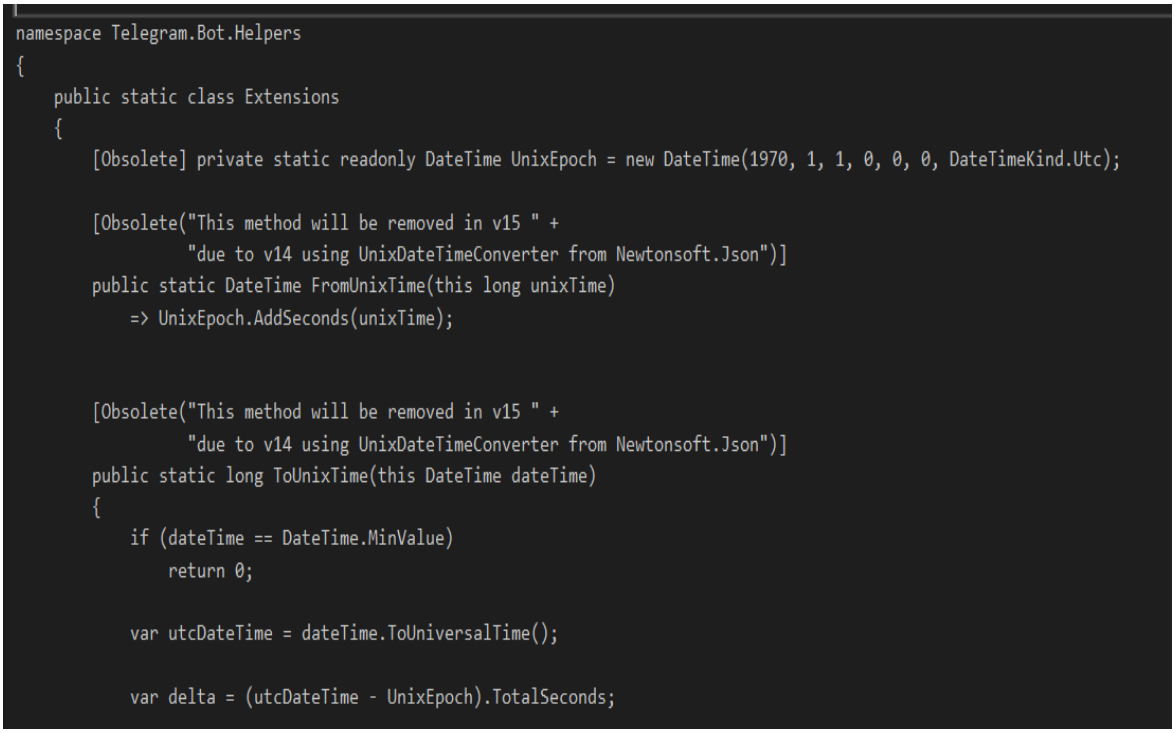
На рисунку 4.6 розглядається реалізація методу Args. Цей метод потрібен для основного функціоналу роботи нашого бота. Він включає в себе реалізацію обміну повідомленнями, розпізнавання на запити певних подій та реалізує охайне формування коду для розробника.



```
namespace Telegram.Bot.Exceptions
{
    public class ApiRequestException : Exception
    {

        public virtual int ErrorCode { get; }


        public ResponseParameters Parameters { get; }


        public ApiRequestException(string message)
            : base(message)
        {
        }


        public ApiRequestException(string message, int errorCode)
            : base(message)
        {
            ErrorCode = errorCode;
        }

        public ApiRequestException(string message, Exception innerException)
            : base(message, innerException)
        {
        }
```

Рисунок 4.8 — Реалізація методу Exeptions

Загалом метод Exeptions більше зроблений для тестування роботи програми. Тобто тестування завантаження файлів або ж обробка невірного запиту.



```
namespace Telegram.Bot.Helpers
{
    public static class Extensions
    {
        [Obsolete] private static readonly DateTime UnixEpoch = new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

        [Obsolete("This method will be removed in v15 " +
                  "due to v14 using UnixDateTimeConverter from Newtonsoft.Json")]
        public static DateTime FromUnixTime(this long unixTime)
            => UnixEpoch.AddSeconds(unixTime);


        [Obsolete("This method will be removed in v15 " +
                  "due to v14 using UnixDateTimeConverter from Newtonsoft.Json")]
        public static long ToUnixTime(this DateTime dateTime)
        {
            if (dateTime == DateTime.MinValue)
                return 0;

            var utcDateTime = dateTime.ToUniversalTime();

            var delta = (utcDateTime - UnixEpoch).TotalSeconds;
```

Рисунок 4.9 — Реалізація методу Helpers

Метод Helpers потрібен для перезавантаження програми у випадку її збою.

```
namespace Telegram.Bot.Requests
{
    /
    [JsonObject(MemberSerialization.OptIn, NamingStrategyType = typeof(SnakeCaseNamingStrategy))]
    public abstract class FileRequestBase<...> : RequestBase<...>
    {

        protected FileRequestBase(string methodName)
            : base(methodName)
        { }


        protected FileRequestBase(string methodName, HttpMethod method)
            : base(methodName, method)
        { }

        protected MultipartFormDataContent ToMultipartFormDataContent(string fileParameterName, InputFileStream inputFile)
        {
            var multipartContent = GenerateMultipartFormDataContent(fileParameterName);

            multipartContent.AddStreamContent(inputFile.Content, fileParameterName, inputFile.FileName);

            return multipartContent;
        }
```

Рисунок 4.10 — Реалізація методу Requests

Тут метод Requests який повністю відповідає за запити користувача до будь- якого телеграм бота. Запит може бути на подобі:

- вислати фотографію;
- вислати файл;
- послати Емодзі;
- пройти тест;

Потрібно розуміти що у телеграмі є своя бібліотека, яка працює по принципу перебірки надісланих файлів користувачем. А у самому методі

Requests є також блок коду ParameterlessRequest.cs, який відповідає за ігнорування полей повідомлення, які не входять до бібліотеки telegram, тим самим не реагуючи на запит і зберігання файлу невірного формату.

```
namespace Telegram.Bot.Types
{

    [JsonObject(MemberSerialization.OptIn, NamingStrategyType = typeof(SnakeCaseNamingStrategy))]
    public class Chat
    {

        [JsonProperty(Required = Required.Always)]
        public long Id { get; set; }

        [JsonProperty(Required = Required.Always)]
        public ChatType Type { get; set; }

        [JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore)]
        public string Title { get; set; }

        [JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore)]
        public string Username { get; set; }

        [JsonProperty(DefaultValueHandling = DefaultValueHandling.Ignore)]
        public string FirstName { get; set; }
```

Рисунок 4.11 — Реалізація методу Helpers Types

Метод Types включає в себе все, що буде знаходитись у нас в програмі. До програми ми можемо додати будь що. В нашому випадку окрім стандартних речей будь якого чат — боту ми в наші аргументи додамо пароль для адміністратора та роботу з опитуванням користувача.

4.5 Створення тесту

Привілегія створення нового тесту буде належати адміністратору програми. Для першого тесту я обрав оцінювання мови програмування C#. Всі питання мені довелося вигадувати самому. На основі знаючого матеріалу з цікавлячої нас теми можна спочатку вибрати список питань(наприклад 25). і потім вибрати кількість варіантів відповідей. Задається адміністратором правильна відповідь, яку при тестуванні не бачить користувач. Потрібно сказати що тут діє відсоткове відношення в залежності від заданого списку питань. Тут спеціальна формула яка реалізована в коді за принципом відсоток правильних відповідей від сумарних питань видає оцінку тесту.

Дуже важливим є те що тут відсутня така функція, як відображення повідомлення на подобі («відповідь не правильна, спробуйте ще раз»), тому що більшість користувачів почнуть відповідати методом так званого «тику». Тому для більшої впевненості вдалого впливу на навчальні результати та, в першу чергу, вдалого оцінювання правильні відповіді будуть показуватись тільки в кінці разом з оцінкою тестування.

Щодо оцінки тестування тут немає числового оцінювання. Якщо користувач набрав оптимальну кількість балів то програма виведе(«отличный результат»), якщо ж користувач набрав малу кількість балів тоді програма виведе («Неплохо, но можно лучше»)

З програмної точки зору ми реалізуемо тест по C# вписуючи вручну всі питання які ми хочемо бачити в ньому. Також повинні вписати варіанти відповідей . Початковий код нашого бота – тестувальника реалізує початкові питання по мові програмування C# рисунок 4.12.

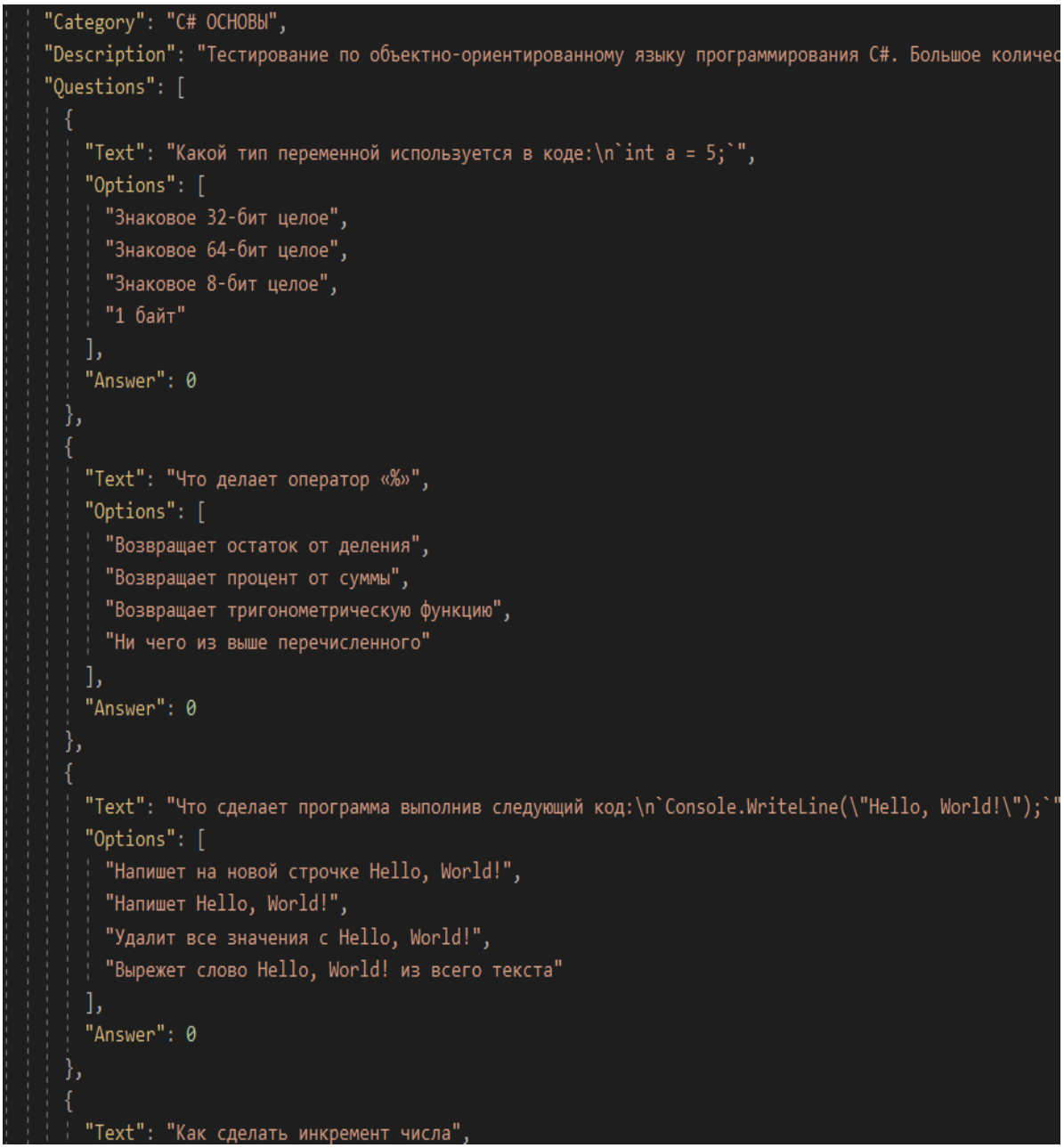


Рисунок 4.12 — Кодова реалізація тесту

Як ми бачимо на рисунку реалізована кодова структура нашого тесту. Загалом програма нараховує список з 25 питань та по 4 варіанти відповіді на них. Правильна відповідь є тільки одна. Все працює по принципу взаємодії з месенджером Telegram. Коли користувач відповідає на питання натискаючи на кнопку відповіді то реалізованний в коді лічільник зараховує правильну відповідь. В кінці в залежності від результату виводить певний коментар.

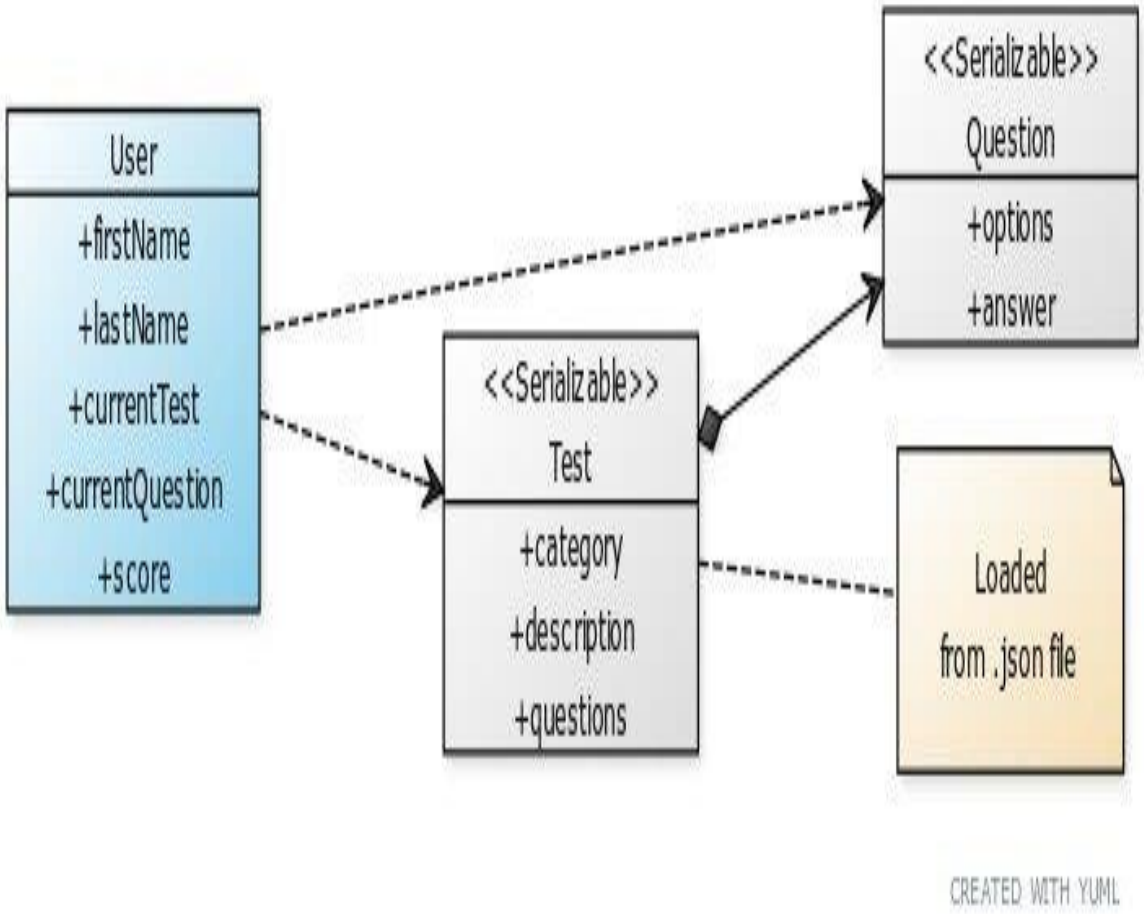


Рисунок 4.13 — Структура додавання тесту

На рисунку 4.13 показана структура додавання нового тесту. Тут діє принцип так званих .json файлів. Тобто це своєрідна хмара звідки основна программа може брати нові тести, які може додати адміністратор. Самий тест як можна побачити синхронізується з реалізованими функціями, які дозволяють візуально зрозуміти роботу бота – тестувальника. Ці функції відповідають за такі речі як:

- авторизація нового користувача;
- видача меню з варіантами вибору категорії тесту;
- видача питання тесту;
- насчитування до лічильника результатів правильних відповідей отриманих від відповіді користувача;

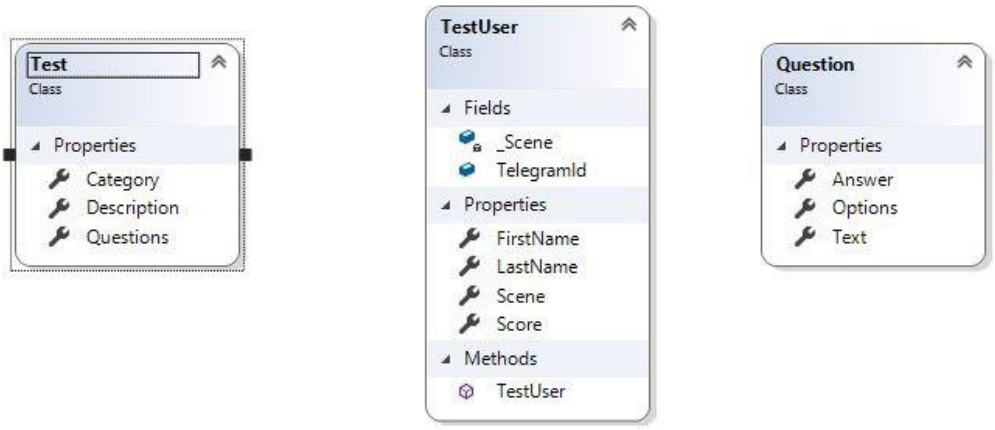


Рисунок 4.14 — Структура логіки тесту

На рисунку 4.14 показана структура логіки тесту та його взаємодія з тестом розробленним окремо. Тут розглядаються окремі класи які дають пряме уявлення про те, як відуально повинна виглядати програма у месенджері Telegram. Тут можна побачити такі реалізовані класи як:

- реалізація старту програми;
- реалізована подача тесту користувачеві;
- реалізовані такі функції відповіді для користувача як батони(кнопки для відповіді, які створенні як основною функцією відповіді та взагалі взаємодією у будь – яких тестувальних ботах Telegram);
- реалізований згідно зі стандартами телеграма видача наступного питання(з коду реалізованного тесту береться питання та відображається у знайомому користувачеві інтерфейсі);

4.6 Використання тесту

Якщо користувач захоче пройти тест достатньо буде зайти в телеграм бот та вибрати потрібну функцію. Чи то правила користування по ньому чи то тест. Якщо був вибраний тест то бот переведе користувача на онлайн тестування.

Проходити сам по собі тест не важко, оскільки програма має простий та звичний для людського ока інтерфейс та гарно працюючий без жодних помилок тест.

В кінці оцінювання знань користувача, за умови авторизації, буде видано ботом коментар про успішність результату.

5 ТЕСТУВАННЯ ПРОГРАМИ

Телеграм доступний для більшості використовуємої техніки. Для тестування необхідно використати стаціонарний комп’ютер або ноутбук чи смартфон на базі одної з операційних систем, що підтримують Телеграм.

Тестування проведено на смартфоні Meizu.m8 із технічними характеристиками:

Екран: 1440x720, мультитач;

Процесор: Mediatek MT6739;

Операційна система: Android;

Оперативна пам'ять: 3 ГБ;

Вбудована пам'ять: 32 ГБ;

Акумулятор: 3200 мАг;

Габарити: 138.3x67.1,x7,1

мм;

Кількість ядер: 4;

Telegram: Telegram v 6.2.

Також із використанням ноутбука з технічними

Екран: 1920x1080 Full HD;

Процесор: Intel HD Graphics 620;

Операційна система: Microsoft Windows 10;

Оперативна пам'ять: 12 ГБ;

Вбудована пам'ять: 256 ГБ;

Telegram: Telegram v 6.2

Оскільки ми вже маемо все необхідне для роботи нашого бота все що нам залишається це знайти його в пошуку телеграм та почати тестування.

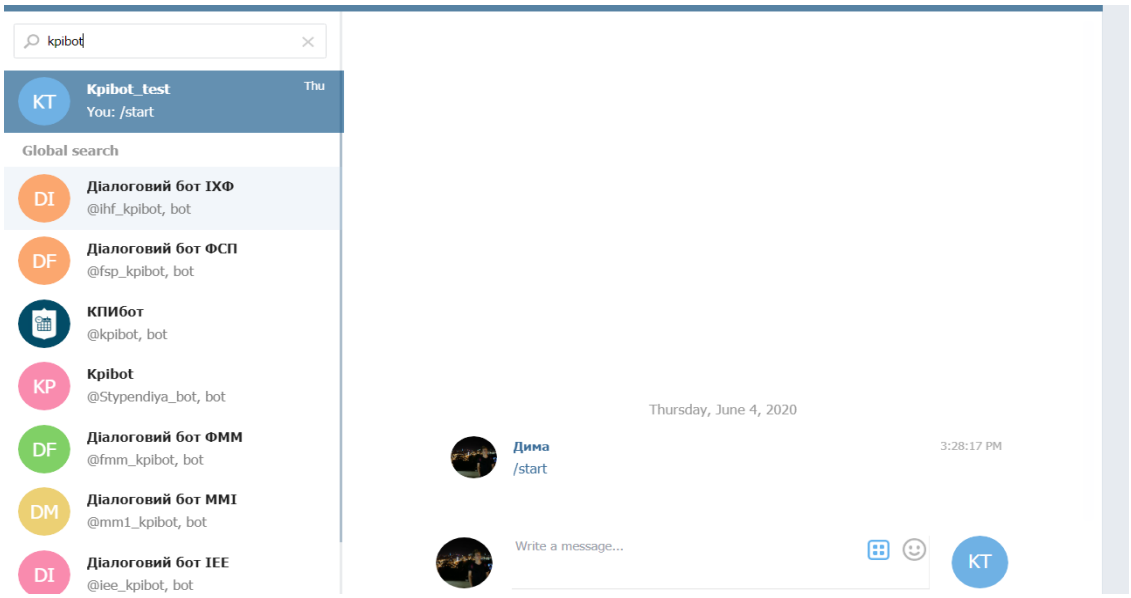


Рисунок 5.1 — Пошук розробленного боту

На рисунку 5.1 показанна адреса нашого бота.

Коли ми знаходимо то прописуємо команду /start

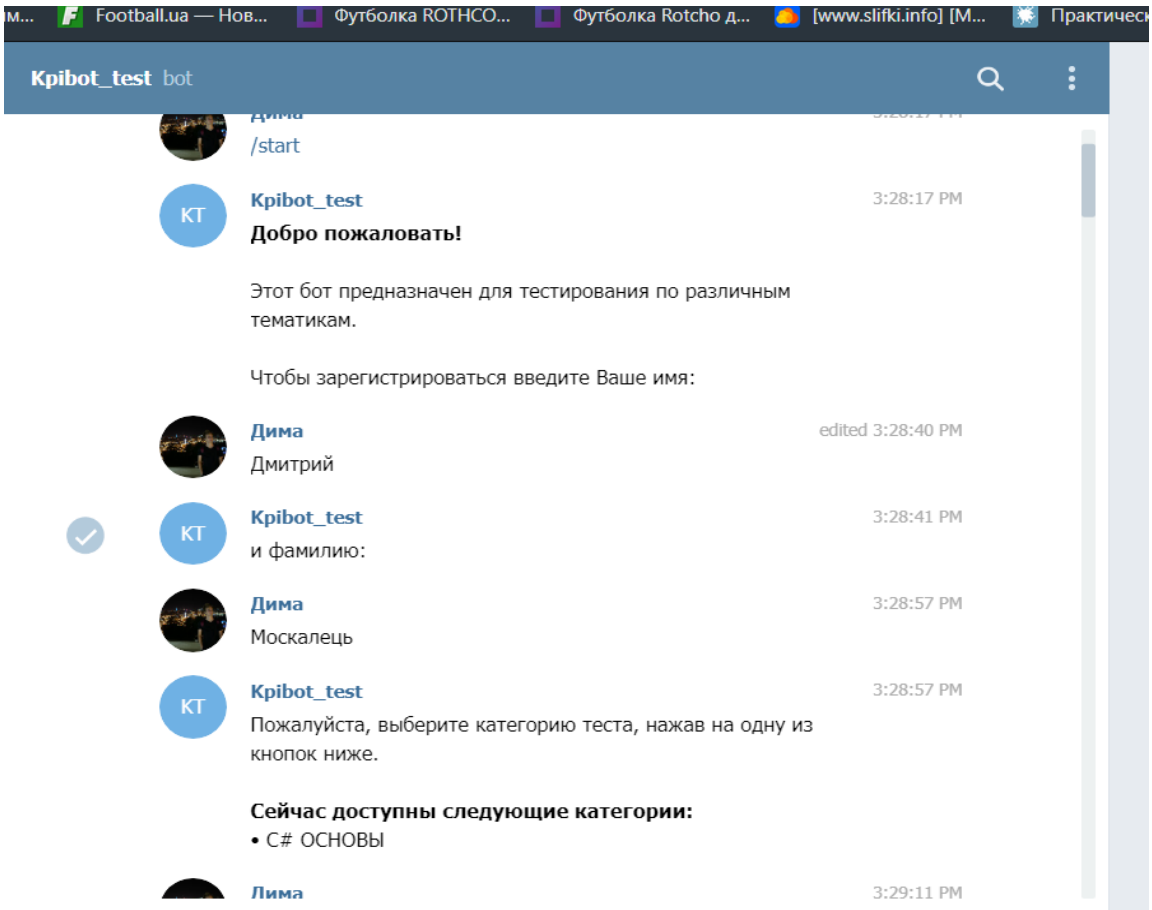


Рисунок 5.2 — Атворизація користувача

Як ми бачимо на рисунку 5.2 після старту роботи бота він просить нас ввеси імя та фамілію. Та ділиться основним функціоналом тобто списком доступних тестів. Потім ми вибираємо кнопки основи C# та розпочинаємо тестування рисунку. 5.3.

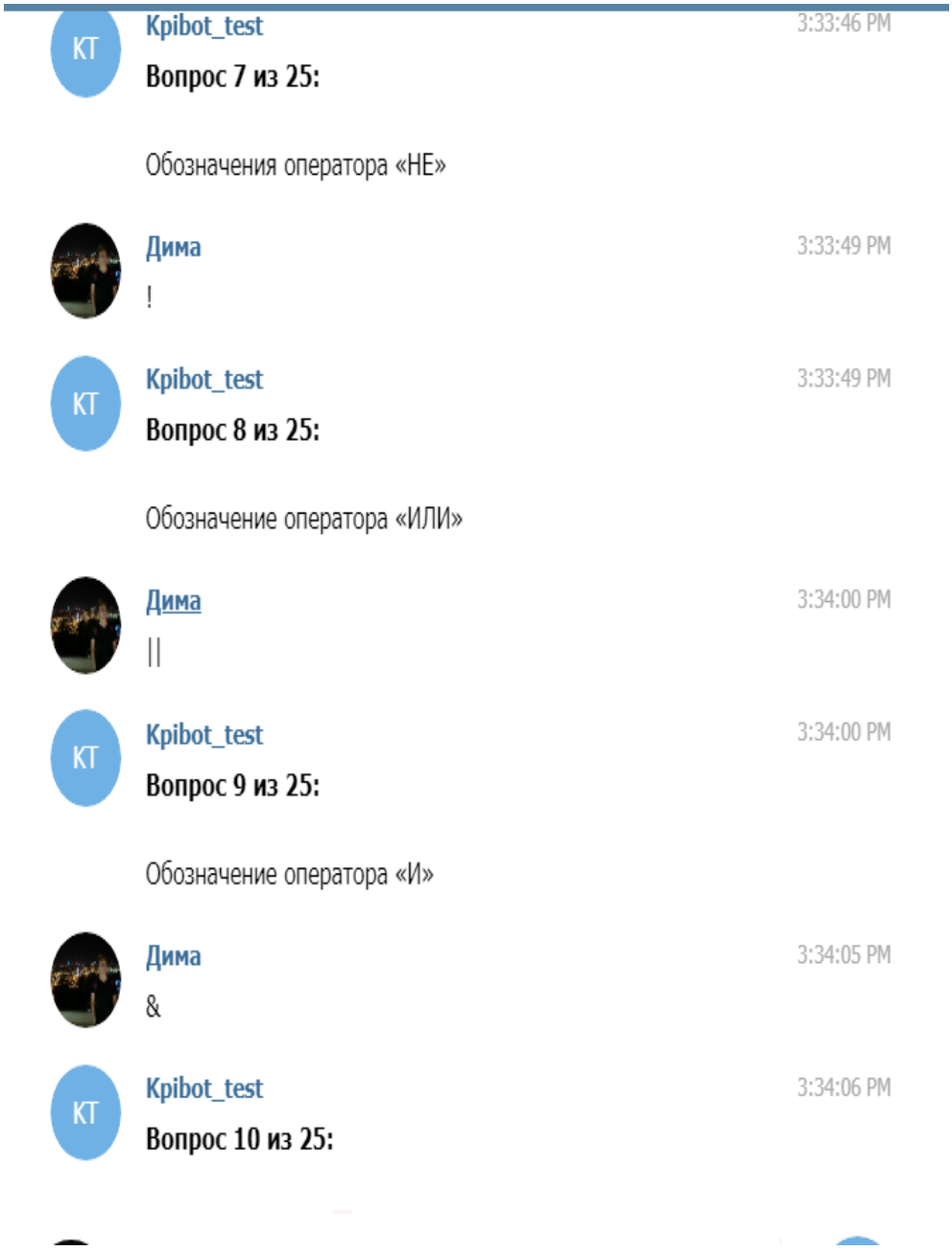


Рисунок 5.3 — Початок тестування

Як ми бачимо є нумерація кожного тесту.

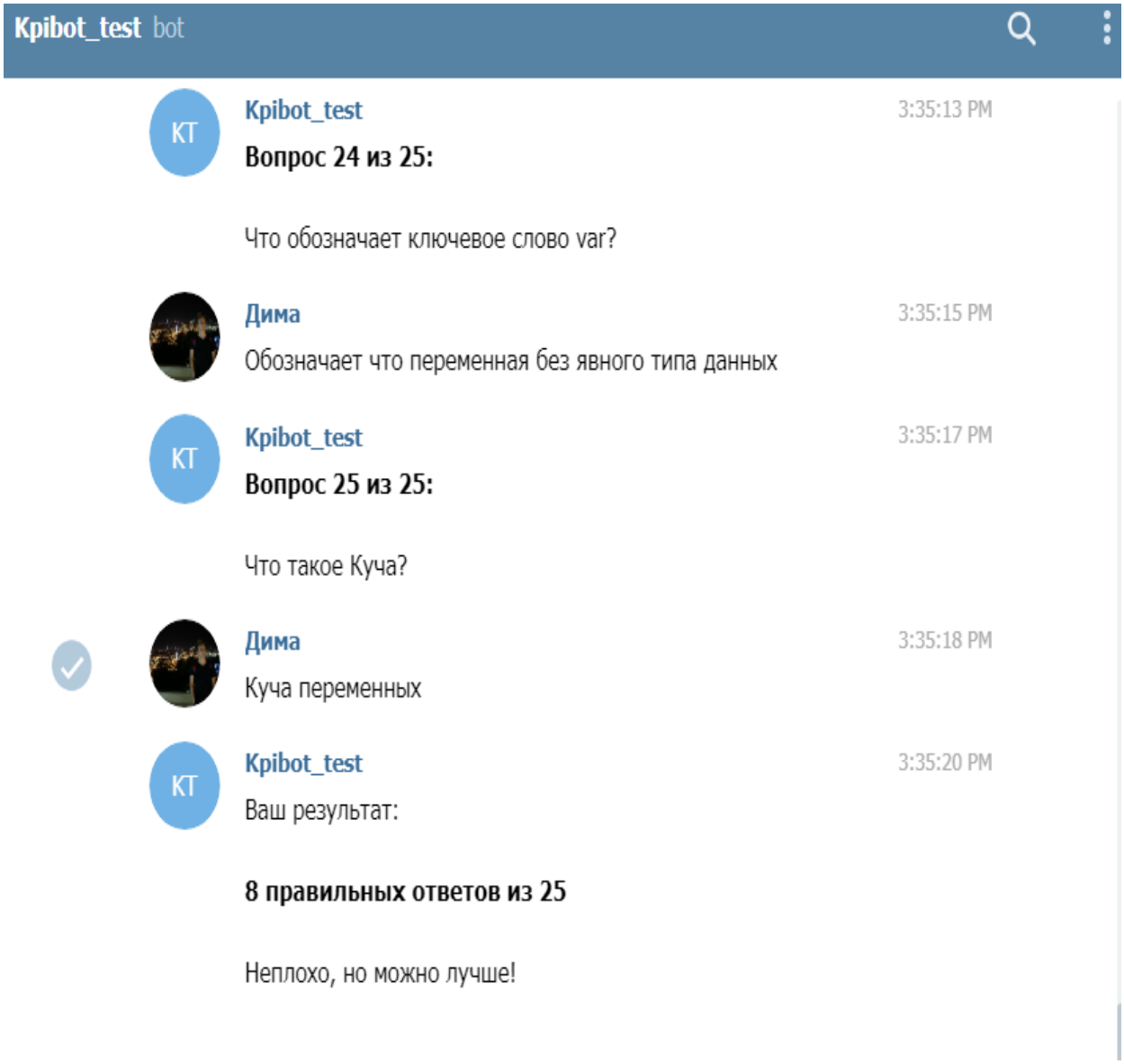


Рисунок 4.4 — Вивід результату

В кінці тесту отримуємо результат, який оцінюється не оцінкою, а коментарем до роботи. Слід зазначити що для іншого тесту оцінка може виступати в цифровому форматі

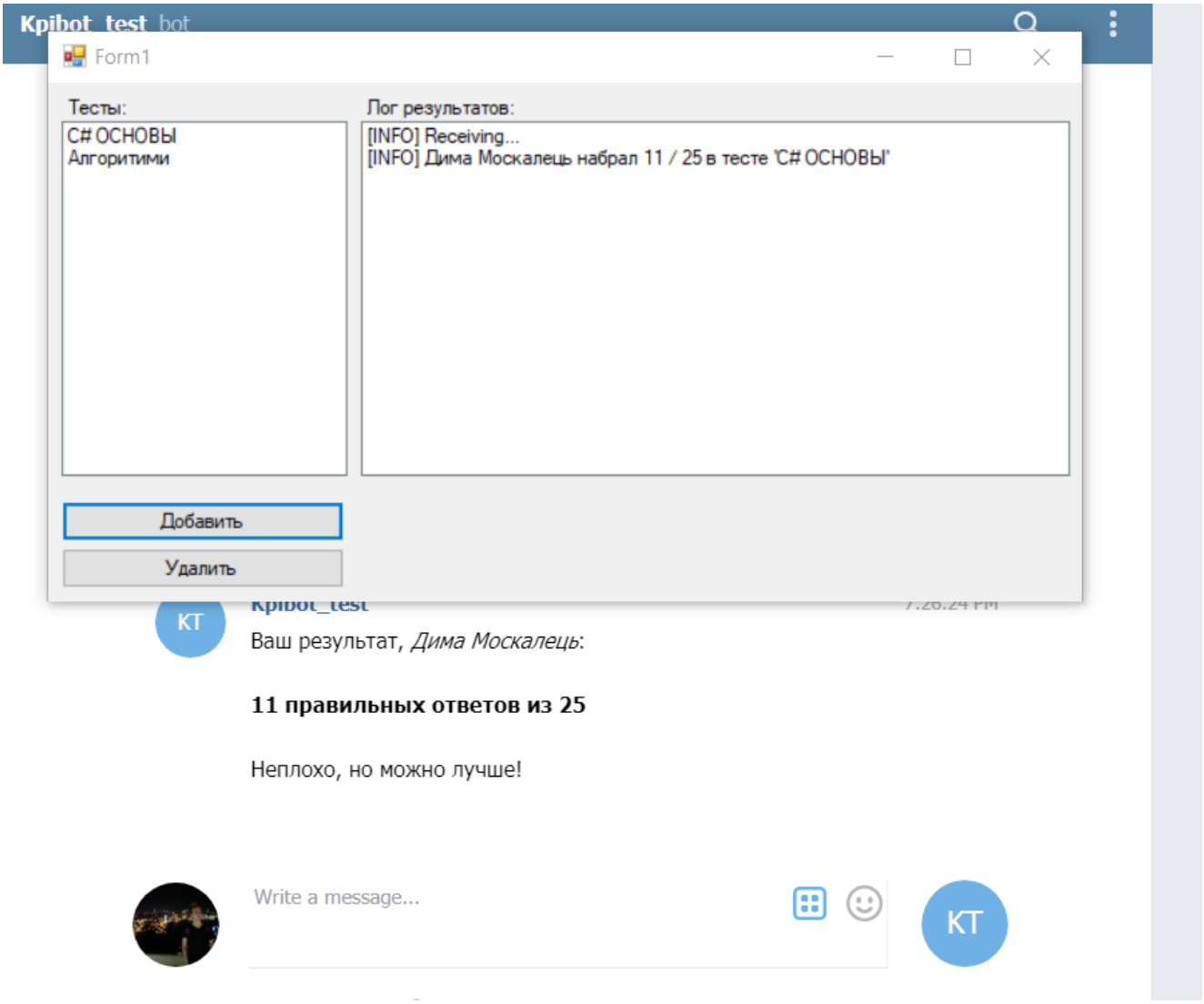


Рисунок 4.5 — Зберігання результату

Всі результати будуть зберігатися у формі 1. Список з результатами усіх авторизованих користувачів буде працювати до тих пір поки адміністратор не вимкне програму.

ВИСНОВКИ

В ході виконання проекту було розроблено сервіс автоматизованного навчання, який дозволить вивчати певний матеріал за допомогою тестування. Які саме питання, залежить від викладача який буде добавляти ці тести.

Було оглянуто певні, поетапні рішення виконання роботи. Також було розглянуто вже існуючи сервіси для автоматизації навчання. Було взято найкращі моменти з цих сервісів та втілено у цей проект.

В кінці було розглянуто що представляє із себе програма. Було зображено роботу програми, візуально яка зображена у інтерфейсі телеграм та інтерфейс збереження результату.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основна інформація про телеграм ботів [Електронний ресурс] – Режим доступу до ресурсу :

   https://greenforest.com.ua/journal/read/14-kanaliv-ta-botiv-u-telegram-dlya-vivchennya-anglijskoi-movi
2. Функціональність телеграм ботів [Електронний ресурс]– Режим доступу до ресурсу :

   https://osvitoria.media/news/zapustyly-zno-bot-chelendzh-v-telegrami-vchyshsya-i-otrymuyesh-pryzy/
3. Основна інформація asp.NET [Електронний ресурс]– Режим доступу до ресурсу :

   https://metanit.com/sharp/aspnet5/1.1.php
4. Microsoft Azure [Електронний ресурс]– Режим доступу до ресурсу: https://prostor/ru//Microsoft_Azure
5. Паттерн Репозиторій принцип дії [Електронний ресурс]– Режим доступу до ресурсу :

   https://bool.dev/blog/detail/pattern-repozitoriy-poeaa
6. Паттерн Репозиторій основна інформація [Електронний ресурс] – Режим доступу до ресурсу :

   https://sohabr.net/habr/post/353840/
7. Паттерн SOLID [Електронний ресурс]– Режим доступу до ресурсу : https://professorweb.ru/my/it/blog/net/solid.php
8. Інформація про паттерни [Електронний ресурс]– Режим доступу до ресурсу : http://cpp-reference.ru/patterns/behavioral-patterns/strategy/

9. Паттерн Стратегія [Електронний ресурс]– Режим доступу до ресурсу : https://ru.wikipedia.org/wiki/Стратегия_(шаблон_проектирования)
10. Паттерн Абстрактна фабрика [Електронний ресурс]. – Режим доступу до ресурсу : https://ru.wikipedia.org/wiki/Абстрактная_фабрика_(шаблон_проектирования)
11. Паттерни, реалізація в ООП [Електронний ресурс]– Режим доступу до ресурсу : https://bool.dev/blog/detail/unit-of-work-patterny-obektno-relyatsionnoy-logiki-poeaa
12. Ілон Маск “Дорога в майбутне”[Література]– Режим доступу до ресурсу: https://www.yakaboo.ua/ilon-mask-tesla-spacex-i-doroga-v-buduschee-1854526.html
13. Вільям Спрінгер “Гид по Computer science для каждого программиста” [Література]– Режим доступу до ресурсу: https://www.piter.com/collection/skoro/product/gid-po-computer-science-dlya-kazhdogo-programmista
14. Роберт Мартин “Идеальный программист” [Література]– Режим доступу до ресурсу: https://n-knigi.com.ua/p674166098-idealnyj-programmist-kak.html?gclid=CjwKCAjwztL2BRATEiwAvnALcjl29k1a7WCE6t3y-CT83GiT0sP1QMFkeCRVTdmZtBlx693g82ntgxoCX84QAvD_BwE
15. Джон Полль Мюллер “Алгоритм для чайников” [Література]– Режим доступу до ресурсу: http://www.dialektika.com/books/978-5-9909446-2-6.html

16. Васильев Алексей Николаевич “ Программирование на C# для начинающих” [Література]– Режим доступу до ресурсу: https://zakazknig.com.ua/programmirovanie-na-c-dlya-nachinaushchikh/?gclid=CjwKCAjwztL2BRATEiwAvnALcht-_Bx8KFWKjKkf8jae5Dcu9k-_WYwlOzkCApUO9tVVWiUi31FoIhoCc-oQAvD_BwE

ДОДАТОК А
Лістнинг програми

```
    using System;
using System.Linq;
using System.Threading;
using System.Windows.Forms;
using TestBot.Forms;

namespace TestBot
{
   static class Program
   {
      [STAThread]
      static void Main(string[] args)
      {
         MyMessages.Current = MyMessages.Rus;

         string token = "1189203787:AAHZN6RHNUprFZsvLTIqVZQ071q5Rf-56ig";

         if (args.Contains("-headless"))
         {
            new MyTestBot(token, new MyConsoleLogger());
            Thread.Sleep(int.MaxValue);
         }
```

```
else
            {
                Application.EnableVisualStyles();
                Application.SetCompatibleTextRenderingDefault(false);
                Application.Run(new MainForm(token));
            }
        }
    }


    public class MyConsoleLogger : IMyLogger
    {
        public void Info(string message)
        {
            Console.WriteLine($"[INFO] {message}");
        }
    }
}
```

```
namespace TestBot
{
    class MyMessages
    {
        public string FormNextQuestionText;
        public string FormNextQuestionOptionFormat;
        public string FormNextQuestionAnswer;
        public string FormNextQuestionNext;
        public string FormNextQuestionDone;
        public string FormNextQuestionCancel;
        public string FormNextQuestionEnterText;
        public string FormNextQuestionEnterOptionFormat;
        public string FormNextQuestionEnterInvalidAnswer;
        public string FormAddTestCategory;
        public string FormAddTestDescription;
        public string FormAddTestNext;
        public string FormAddTestCancel;
        public string FormAddTestEnterCategory;
        public string FormAddTestEnterDescription;
          public string FormMainTests;
          public string FormMainResultLog;
          public string FormMainAdd;
          public string FormMainRemove;
          public string FormMainSelect;
          public string FormMainErrorLoading;
                public string FormMainErrorSaving;
```

```
public string SceneStartText;
public string SceneStartSurname;

public string SceneMenuText;

public string SceneDescriptionTextFormat;
public string SceneDescriptionContinue;

public string SceneQuestionTextFormat;
public string SceneQuestionLogFormat;

public string SceneResultTextFormat;
public string SceneResultTextGood;
public string SceneResultTextNotBad;
public string SceneResultMenu;

public static readonly MyMessages Rus = new MyMessages
{
   FormNextQuestionText = "Текст вопроса:",
   FormNextQuestionOptionFormat = "Вариант ответа {0}:",
   FormNextQuestionAnswer = "Номер правильного ответа ({0}-{1}):",
   FormNextQuestionNext = "Далее",
   FormNextQuestionDone = "Готово",
   FormNextQuestionCancel = "Отмена",
   FormNextQuestionEnterText = "Введите текст вопроса",
   FormNextQuestionEnterOptionFormat = "Введите вариант ответа {0}",
            FormNextQuestionEnterInvalidAnswer = "Неверный номер правильного
      ответа",
```

```
        FormAddTestCategory = "Категория:",
        FormAddTestDescription = "Описание/дополнительная информация:",
        FormAddTestNext = "Далее",
        FormAddTestCancel = "Отмена",
        FormAddTestEnterCategory = "Введите категорию",
        FormAddTestEnterDescription = "Введите описание",

        FormMainTests = "Тесты:",
        FormMainResultLog = "Лог результатов:",
        FormMainAdd = "Добавить",
        FormMainRemove = "Удалить",
        FormMainSelect = "Выберите тест который хотите удалить",
        FormMainErrorLoading = "Произошла ошибка во время инициализации
бота. Приложение будет закрыто",
        FormMainErrorSaving = "Произошла ошибка во время сохранения",

        SceneStartText = "*Добро пожаловать!*\n\n" +
            "Этот бот предназначен для тестирования по различным тематикам.\n\n"
+
            "Чтобы зарегистрироваться введите Ваше имя:",
        SceneStartSurname = "и фамилию:",
       SceneMenuText = "Пожалуйста, выберите категорию теста, нажав на одну
из кнопок ниже.\n\n" +
            "*Сейчас доступны следующие категории:*\n",

        SceneDescriptionTextFormat = "*Информация:*\n\n{0}\n\nНажмите кнопку
*Продолжить*, когда будете готовы начать",
        SceneDescriptionContinue = "Продолжить",
```

```
        SceneQuestionTextFormat = "*Вопрос {0} из {1}:*\n\n{2}",
        SceneQuestionLogFormat = "{0} {1} набрал {2} / {3} в тесте '{4}'",

        SceneResultTextFormat = "Ваш результат, _{0} {1}_:\n\n*{2} правильных ответов из {3}*\n\n",
        SceneResultTextGood = "Отличный результат!",
        SceneResultTextNotBad = "Неплохо, но можно лучше!",
        SceneResultMenu = "В меню",
    };

    public static MyMessages Current { get; set; }
  }
}
```

```
        using System;
using System.Collections.Generic;
using Telegram.Bot;
using Telegram.Bot.Args;
using Telegram.Bot.Types.Enums;
using TestBot.Models;

namespace TestBot
{
    public class MyTestBot
    {
        public static MyTestBot Instance { get; private set; }

        public readonly List<Test> Tests;
        public readonly Dictionary<int, TestUser> Users;
        public readonly TelegramBotClient Client;
        public readonly IMyLogger Logger;

        public MyTestBot(string token, IMyLogger logger)
        {
            if (Instance != null) throw new InvalidOperationException();
            Instance = this;

            Tests = Test.Load();
            Users = new Dictionary<int, TestUser>();
            Client = new TelegramBotClient(token);
            Logger = logger;
```

```
            Client.OnMessage += Bot_OnMessage;
            Client.StartReceiving();

            Logger.Info("Receiving...");
        }

        private void Bot_OnMessage(object sender, MessageEventArgs e)
        {
            if (e.Message.Chat.Type == ChatType.Private && e.Message.Text != null)
            {
                if (!Users.TryGetValue(e.Message.From.Id, out TestUser user))
                {
                    user = new TestUser(e.Message.From.Id);
                    Users.Add(e.Message.From.Id, user);
                }
                else
                {
                    user.Scene.Accept(user, e.Message.Text);
                }
            }
        }
    }

    public interface IMyLogger
    {
        void Info(string message);
    }
}
```